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

Agentes de Mineração e sua Aplicação no Domínio de Auditoria Governamental

Carlos Vinícius Sarmento Silva

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2011

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

Coordenador: Prof. Dr. Mauricio Ayala Rincón

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof. Dr. Wagner Meira Junior — DCC/UFMG
Prof. Dr. Jacir Luiz Bordim — CIC/UnB

CIP — Catalogação Internacional na Publicação

Silva, Carlos Vinícius Sarmento.

Agentes de Mineração e sua Aplicação no Domínio de Auditoria Governamental / Carlos Vinícius Sarmento Silva. Brasília : UnB, 2011.
122 p. : il. ; 29,5 cm.

Tese (Mestrado) — Universidade de Brasília, Brasília, 2011.

1. Mineração de Dados, 2. Sistemas Multiagentes, 3. Auditoria,
4. AGMI, 5. e-gov

CDU 004.8

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

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

Agentes de Mineração e sua Aplicação no Domínio de Auditoria Governamental

Carlos Vinícius Sarmento Silva

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof. Dr. Wagner Meira Junior
DCC/UFMG

Prof. Dr. Jacir Luiz Bordim
CIC/UnB

Prof. Dr. Mauricio Ayala Rincón
Coordenador do Mestrado em Informática

Brasília, 11 de Março de 2011

Dedicatória

À *Tayana*,
que tem regado meu coração com seu *calmo amor prestante*,
sem se cansar de fazer cócegas na minha alma com suas doces risadas.

Agradecimentos

G. K. Chesterton escreveu certa vez que o teste de toda felicidade é a gratidão. “Nós agradecemos às pessoas pelos chinelos e charutos que ganhamos nos nossos aniversários. Por que não posso agradecer a alguém pelo presente de ter nascido?”. Desta forma, dirijo as primícias de minha gratidão à Ele, cujas palavras nos meus dias de inquietude sempre foram: “Vinde a mim, todos os que estais cansados e sobrecarregados, [...] e achareis descanso para as vossas almas”.

À Prof. Célia pela orientação, parceria e alegre otimismo, que me contagiou antes mesmo de concebermos este projeto. Ao estimado colega e Gerente na Diretoria de Informações Estratégicas da SPCI/CGU, Henrique Rocha, que acreditou na idéia, contribuindo para concretização da mesma. Ao Diretor de Sistemas e Informação da CGU, José Geraldo Loureiro Rodrigues, e ao Coordenador-Geral de Informação, Oswaldo Iglesias de Azeredo, que me apoiaram nesse projeto, permitindo assim a conclusão deste mestrado. E também a todos os demais colegas da CGU que direta ou indiretamente ajudaram nesse projeto.

Finalmente, àqueles que habitam no recôndito do meu coração, a saber, meus queridos pais Carlos e Graça, meu amado irmão Apolo, e todos os meus bons e velhos amigos que estiveram ao meu lado durante todo esse tempo.

Resumo

O trabalho de auditoria governamental tem sido realizado no âmbito do Poder Executivo Federal pela Controladoria-Geral da União. Várias estratégias são utilizadas visando a prevenção e o combate à corrupção. No entanto, algumas atividades tais como detecção de cartéis em licitações são limitadas devido à complexidade de se correlacionar informações para geração de conhecimento útil para os auditores através da análise de bases de dados. A área de Mineração de Dados tem sido alvo de várias pesquisas tendo bons resultados no processo de descoberta de conhecimento em grandes bases de dados onde várias técnicas já foram definidas nesta área tais como classificação, clusterização e regras de associação. Sistema Multiagentes por sua vez, apresenta consideráveis vantagens no sentido de possibilitar a distribuição do processamento e fazer uso de autonomia de agentes de softwares para realização de tarefas complexas. Essas duas áreas de estudo, até recentemente separadas, são integradas neste trabalho através de *AGent Mining Integration* (AGMI), uma arquitetura que integra diferentes técnicas de mineração de dados utilizando uma abordagem multiagentes para automatização do processo de descoberta de conhecimento. AGMI é composta por agentes que operam em três diferentes camadas: estratégica, tática e operacional. Através da autonomia de agentes, AGMI é capaz de integrar técnicas de mineração de dados de forma distribuída e utilizar heurísticas para melhoramento do conhecimento encontrado. Neste trabalho é apresentado um protótipo do AGMI que foi testado com dados reais de licitações extraídas do Sistema *ComprasNet*. Vários experimentos foram realizados explorando os aspectos de distribuição do processamento e autonomia dos agentes. AGMI apresentou bons resultados quanto ao desempenho, capacidade autônoma de melhorar o conhecimento descoberto e quanto à qualidade do conhecimento apresentado. Comparando com a abordagem testada, utilizando apenas o algoritmo de Regras de Associação, os experimentos com AGMI mostraram um aumento de 170% na qualidade média das 10 melhores regras encontradas e de 350% na qualidade média das 100 melhores regras encontradas. Além disso, AGMI aumentou a qualidade de 193 regras, através de heurística aplicada autonomamente pelo agente Avaliador. As regras descobertas nos experimentos foram analisadas por especialistas da Controladoria-Geral da União e apresentaram fortes indícios de irregularidades em licitações tais como cartéis, simulação de concorrência e direcionamento de editais.

Palavras-chave: Mineração de Dados, Sistemas Multiagentes, Auditoria, AGMI, e-gov

Abstract

In Brazil, government auditing is performed by the Office of the Comptroller General (CGU), where several approaches are being used to prevent and fight corruption. However, some activities such as government purchasing fraud detection are limited by the difficulty in finding effective ways to implement. The main problem focused by this research project is how to extract and generate useful knowledge from huge databases of Brazilian Federal procurement processes, in order to help the governmental auditing work. However, activities like detection of cartels are a complex problem in many senses. In terms of finding useful auditing knowledge, because of the volume of data to correlate information, and also because of the dynamism and diversified strategies used by companies to hide their fraudulent operations. In this research, we combine two originally separated areas and increasingly interrelated: distributed multi-agent systems and data mining. In our approach, we prove the interaction features in a bilateral and complementary way, by introducing AGMI - an AGent-MIning tool for automate knowledge discovery process in a distributed way. Considering the data mining perspective, we have used different model functions, such as clusterization and link analysis with association rules. Autonomous agents are also used in the process in order to improve the discovered knowledge quality. To validate the usage of AGMI, we have performed several experiments using real data from ComprasNet, a government purchasing system of Brazil. Our approach resulted in expressive discovered knowledge. Considering a tested approach using only Association Rule algorithm, the AGMI's experiments have shown a rule quality improvement of 170% in the top 10 rules and 350% in the top 100 rules. Besides, AGMI also enhanced the quality of 193 rules through the autonomous heuristics of Evaluator Agent. According to the auditing experts, the discovered knowledge shall help the work of the CGU auditors in the detection, prevention and monitoring of cartels acting in public procurement processes.

Keywords: Data-Mining, MAS, AGMI, Auditing, e-gov

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

DMA Framework *Data Mining for Agents Framework*. 53, 55, 56, 60, 61, 66, 78–80, 95

EM *Expectation Maximization*. 10–13, 50, 72, 78–80

RQ *Rule Quality*. 85, 87, 90–92

AGMI *AGent Mining Integration*. 5, 55–58, 60–65, 67, 72, 76–78, 83–89, 91, 92, 94–96

AMII *Agent-Mining Interaction and Integration*. 2, 4, 17, 44, 95

AMS *Agent Management System*. 106, 107

ARFF *Attribute-Relation File Format*. 104

CADE Conselho Administrativo de Defesa Econômica. 40, 42

CGU Controladoria-Geral da União. 1, 2, 4, 35, 36, 41–43, 45, 91, 95

ComprasNet Portal de Compras do Governo Federal. 5, 37, 45, 47, 77, 95, 96

CRISP-DM *CRoss-Industry Standard Process for DM*. 7, 8, 45

CT *Containers Table*. 106

DCBD Descoberta de Conhecimentos em Base de Dados. 2, 4, 6–8, 10, 32, 91, 94, 110

DF *Directory Facilitator*. 61, 106, 107

FIPA *The Foundation for Intelligent Physical Agents*. 22, 25, 30, 105–107

FIPA-ACL *FIPA Agent Communication Language*. 22, 23, 65, 67

GADT *Global Agent Descriptor Table*. 106

GNU *General Public Licence*. 103

IA Inteligência Artificial. 17

IAD Inteligência Artificial Distribuída. 17

JADE *Java Agent Development Framework*. 4, 25, 31, 44, 55, 60, 61, 65–69, 72, 78, 95, 105–110

JVM *Java Virtual Machine*. 63, 72

KIF *Knowledge Interchange Format*. 22

KQML *Knowledge Query and Manipulation Language*. 22

KSE *Knowledge Sharing Effort*. 22

MD Mineração de Dados. 2–11, 15, 21, 24, 31, 32, 42–44, 46, 49, 53–63, 65–70, 72–74, 76–83, 85–89, 91, 95, 97, 103, 110

OCDE Organização para a Cooperação e Desenvolvimento Econômico. 40

OMG *Object Management Group*. 30

ONU Organização das Nações Unidas. 33

OWL *Web Ontology Language*. 25

SIAFI Sistema Integrado de Administração Financeira do Governo Federal. 1

SISG Sistema de Serviços Gerais. 37

SMA Sistemas Multiagentes. 2–5, 17, 20, 21, 24–26, 31, 32, 42, 43, 53, 78, 83, 95, 105, 106, 110

TCU Tribunal de Contas da União. 41

UML *Unified Modeling Language*. 28, 30, 31

W3C *World Wide Web Consortium*. 25

Capítulo 1

Introdução

Atualmente o volume de dados produzidos e armazenados pelos diversos sistemas de computação tem aumentado expressivamente. A informatização dos diversos setores do mercado e também do governo tem sido a causa primária deste aumento na produção de dados digitais. A Administração Pública Brasileira atual mantém a maioria de seus processos apoiados por sistemas computacionais. Desde 1987, com a criação do Sistema Integrado de Administração Financeira do Governo Federal (SIAFI), o governo brasileiro tem usado operações eletrônicas para gerenciamento de sua contabilidade. O SIAFI, por ser usado em todo o Governo Federal, é um bom exemplo para demonstrar essa explosão de crescimento de dados no âmbito governamental. Podemos encontrar estatísticas sobre o uso do SIAFI no site da Secretaria do Tesouro Nacional (Secretaria do Tesouro Nacional, 2011). Como exemplo, somente no ano de 1998, a média mensal de transações executadas no SIAFI era de aproximadamente 39.000.000, entre elas, consultas e inserção de dados. Em 2002, essa média saltou para 63.800.000, e em 2009 para 87.000.000 de transações mensais, com picos de 134.000.000 de transações no mês de dezembro. Isso mostra que a utilização desse sistema aumentou aproximadamente 123% em 10 anos. Ainda em 1999, a média de acessos simultâneos no SIAFI era de 1.703 usuários. Esse número praticamente dobra em 10 anos atingindo uma média de 3.323 acessos simultâneos. No que tange à inserção de dados no SIAFI, só no ano de 2004, foram registrados 20.442.295 de documentos. Esse número de registros subiu para 31.461.897 no ano de 2009, levando em conta os registros anteriores, os quais permanecem como informações históricas no sistema.

Outro sistema, que reflete também o crescimento exponencial do número de informações nos âmbito do Governo Federal é o Portal da Transparência, criado e mantido pela Controladoria-Geral da União (CGU), que atualmente mantém mais de um bilhão de registros que totalizam cerca de 7,6 trilhões de reais em gastos do Governo Federal, tais como transferências de recursos, gastos diretos e cartões corporativos conforme apresentado na Tabela 1.1 `www.portaltransparencia.gov.br`.

1.1 Caracterização do Problema

Os dados provenientes desses e de outros sistemas de informação do governo são utilizados pelos órgãos de auditoria governamental para planejamento e execução de suas auditorias e fiscalizações dos recursos públicos. No âmbito do Poder Executivo Federal,

Tabela 1.1: Dados do Portal da Transparência extraídos do site no dia 7/12/2010

Informações gerais	
Recursos Envolvidos (mensais)	R$ 7.616.861.367.877,93
Informações Registradas	1.033.287.721
Portadores de Cartão de Pagamento do Governo Federal	13.462
Favorecidos	
Pessoas Físicas (incluindo os programas sociais)	19.759.966
Pessoas Físicas (excluindo os programas sociais)	1.894.580
Pessoas Jurídicas	360.365
Programas de Governo	
Total de beneficiários	17.865.386
Total de programas	536
Total de ações de governo	5.288

a CGU têm direcionado esforços no sentido de utilizar tecnologias em análises de dados para desenvolvimento de ações voltadas à promoção da transparência e à prevenção da corrupção.

No entanto, a maior dificuldade se encontra no correlacionamento das informações para geração de conhecimento útil para os auditores. Desta forma, as alternativas atualmente se restringem a consultas aos sistemas em casos pontuais ou preparação de amostras estatísticas que diminuem o universo para um conjunto reduzido de informações, proporcional à capacidade operacional do Órgão. Perde-se assim, o valor de se armazenar todos esses dados, pois este valor depende da habilidade de se extrair conhecimento útil dos dados para que de alguma forma seja agregado algo ao patrimônio da corporação, seja financeiro ou intelectual.

1.2 Justificativa

Objetivando lidar com grandes volumes de dados, a utilização de técnicas de Mineração de Dados (MD) tem se mostrado de grande valia na obtenção de informações e no processo de descoberta de conhecimento (Witten e Frank, 2005). Estas técnicas pertencem a um ramo da Ciência da Computação conhecido como Descoberta de Conhecimentos em Base de Dados (DCBD), o qual associado à sub área de Sistemas Multiagentes (SMA) tem se apresentado como abordagem útil no processamento distribuído de grandes bases de dados com uso de agentes de software de MD.

Na última década, SMA e MD tem surgido como duas áreas de pesquisa fortemente inter-relacionadas, abrindo espaço para o campo de pesquisa de interação e integração entre agentes e mineração (*Agent-Mining Interaction and Integration* (AMII)). Este novo campo tem unido esforços de ambos os lados para reunir os benefícios de se unir as duas áreas (Ralha, 2009).

Voltando ao problema de correlacionamento de informações no âmbito do Governo Federal, um dos problemas encontrados nos trabalhos de auditoria, em análises de processos de licitação, é o chamado Rodízio de Licitações. Empresas que se juntam em cartéis para superfaturar as compras do governo, trazendo enormes prejuízos para a Administração

Pública. A prática é ilegal e a pena pode chegar a 5 anos de reclusão. No entanto, a detecção dos grupos suspeitos de praticar rodízio de licitação é bastante difícil.

A análise dos possíveis grupos de empresas praticarem cartel num banco de dados é um problema que envolve o levantamento de todas as possíveis combinações das empresas com grupos de pelo menos dois participantes. Um algoritmo de força bruta para levantamento desses possíveis grupos seria de ordem exponencial $O(2^n)$, como pode ser observado a seguir:

$$\sum_{i=2}^{n} C_i^n = 2^n - 1$$

onde n é o número total de empresas num banco de dados

Dessa forma, não há solução determinística que auxilie eficazmente a tarefa de detecção de cartéis em licitações, sendo feita de forma geral apenas através de denúncias. Neste sentido, o uso de agentes de MD mostra-se de fundamental importância no trabalho de auditoria governamental. Entretanto, tarefas como preparação de *datasets*, execução de diferentes algoritmos de MD e avaliação de conhecimento encontrado são adequadas para integração com SMA, considerando os aspectos de distribuição, automatização de tarefas ou execução de algoritmos em paralelo. Assim, este trabalho tem como objetivo propor uma solução que auxilie na detecção de cartéis em licitações de forma eficiente, automatizando a maior quantidade de tarefas possíveis nas fases de MD.

1.3 Premissadas Adotadas

Para definir melhor o escopo deste trabalho, algumas premissas foram adotadas relativas ao domínio do problema, às limitações da solução proposta e ao grau de investigação das áreas de conhecimento estudadas. Essas premissas são listadas a seguir:

- A detecção de cartéis em licitações e na maioria dos casos uma tarefa difícil, sendo assim, inviável sem o auxílio computacional
- A avaliação do conhecimento descoberto para auxílio ao problema de detecção de cartéis em licitações deve ser realizada com ajuda de especialista em auditoria pública.
- A abordagem adotada nesse trabalho não é universal em sua completude. Portanto, não há garantia que o modelo proposto detecte todos os grupos de empresas que praticaram cartéis em licitações.
- A solução adotada utilizando integração de técnicas de MD é uma abordagem inovadora descoberta empiricamente, e que trouxe bons resultados experimentais. Possivelmente outras abordagens podem ser criadas para auxílio do problema. No entanto, nenhuma foi encontrada na literatura pesquisada.
- Esta pesquisa é um trabalho inicial de junção de duas áreas definidas isoladamente. Assim, não será escopo do trabalho o aprofundamento em ambas as áreas.
- Algumas especificações de parâmetros utilizados neste trabalho foram realizadas com base no conhecimento do auditor especialista.

- Não foi escopo deste trabalho discutir as características de sistemas distribuídos, tais como comunicação, distribuição de tarefas, alocação de recursos, balanceamento de carga, entre outros. Embora ambas as áreas de investigação da pesquisa possibilite uma discussão mais profunda sobre sistemas distribuídos, nos ateremos apenas à especificação de um modelo que possibilite distribuição do processamento de forma paralela, e a alguns testes exemplificando a funcionalidade do mesmo.

1.4 Contribuição

O objetivo principal deste trabalho é propor uma arquitetura de AMII e implementar um protótipo para descoberta de conhecimentos validado no contexto de auditoria governamental, o qual seja passível de uso em outros domínios de conhecimento. As técnicas de MD utilizadas serão executadas por agentes de software para que estes possam trocar informações entre si no intuito de enriquecer o conhecimento encontrado, utilizando as vantagens de um processamento distribuído. O protótipo foi validado utilizando uma base real de licitações públicas para auxílio na solução do problema de detecção de cartéis em licitações.

Como objetivos secundários deste trabalho podemos citar:

- Estudo de duas áreas de Computação até o momento desassociadas – SMA e MD – investigando a área de pesquisa de DCBD;
- Estudo e adaptação de algoritmos de MD disponíveis na ferramenta Weka, e desenvolvimento de algoritmos para possibilitar o uso em ambientes de SMA;
- Estudo e uso do framework de desenvolvimento de SMA denominado *Java Agent Development Framework* (JADE);
- Automatização de tarefas dispendiosas como a preparação de *datasets* para os algoritmos de MD;
- Análise e implementação de autonomia em agentes do SMA;
- Interpretação e avaliação do conhecimento, através de regras utéis aos auditores, para a detecção de cartéis em licitações na CGU.

1.5 Metodologia

Inicialmente, foi feita a revisão da literatura relativa às áreas envolvidas neste trabalho, considerando a área de AMII. O Capítulo 2 apresenta o estudo das áreas de MD, DCBD e SMA, feito no intuito de identificar as possíveis formas de integração das mesmas. A área de Auditoria Governamental foi estudada e apresentada no Capítulo 3 para que se pudesse conhecer melhor o contexto de aplicação desta pesquisa. Através da revisão da literatura de MD, DCBD, SMA e Auditoria Governamental, foram identificados, ainda no Capítulo 3, os principais desafios para apresentação da proposta de solução, com referência a alguns trabalhos correlatos.

A partir desses estudos, apresentamos no Capítulo 4 a proposta de solução para o problema de cartéis em licitações. Inicialmente foi proposta uma solução baseada na

integração de técnicas de MD, a qual foi testada. Com base nessa solução, foi projetada uma arquitetura de integração de MD e SMA, denominada *AGent Mining Integration* (AGMI) para automatização do processo e enriquecimento do conhecimento encontrado mediante a utilização de agentes de software autônomos. Um protótipo da AGMI foi utilizado para execução de testes e comparação dos resultados, com bases reais de licitação do Portal de Compras do Governo Federal (ComprasNet).

No Capítulo 5, apresentamos os experimentos executados desde a abordagem preliminar até a proposta final do modelo comparando os resultados e mostrando a evolução da pesquisa. É também apresentado nesse capítulo o conhecimento descoberto através da utilização do modelo proposto.

Por fim, as Conclusões e os Trabalhos Futuros são discutidos no Capítulo 6.

Capítulo 2

Fundamentos Teóricos

Este capítulo apresenta os fundamentos teóricos utilizados neste trabalho de pesquisa. A Seção 2.1 apresenta as definições, conceitos e técnicas relacionados à atividade de descoberta de conhecimentos utilizando MD. Na Seção 2.2 são apresentados os conceitos relacionados a agentes e SMA, e os assuntos relacionados ao tema, que serão utilizados na proposta deste trabalho.

2.1 Mineração de Dados e Descoberta de Conhecimento

Nos últimos anos, a quantidade de dados produzida por computadores tem crescido acentuadamente. Este fato pode ser evidenciado claramente na quantidade de informações disponíveis e compartilhadas na Internet. Neste cenário, grandes bases de dados se tornaram comuns. Podemos citar as bases de dados de instituições financeiras, cartões de créditos, hospitais e governo.

Na definição de Frawley et al. (1992), DCBD é uma extração não trivial de informações implícitas, previamente desconhecidas e potencialmente úteis de uma base de dados. O processo é classificado como não trivial porque envolve decisões que estão além da aplicação das técnicas. Por exemplo, a definição do problema, para que seja possível encontrar um caminho de otimização através da aplicação correta de algoritmos para extração da informação.

Outra definição de DCBD encontra-se em Fayyad et al. (1996), onde DCBD é um processo de identificação não trivial de padrões novos, válidos, potencialmente úteis e compreensível nos dados. Chama-se de processo pelo fato de DCBD ser realizado em vários passos, desde a preparação dos dados, até a avaliação dos padrões encontrados, podendo passar por outros refinamentos. As características de novidade, utilidade e compreensibilidade do padrão encontrado remete especialmente ao conceito de conhecimento descoberto.

Ainda segundo Fayyad et al. (1996), as noções de novidade e compreensibilidade dos padrões encontrados podem ser muito subjetivas, entretanto, a compreensibilidade pode ser estimada por sua simplicidade, por exemplo, o tamanho em bits do padrão encontrado. Já o conceito de utilidade pode ser medido em certos casos pelo ganho que se teve com o conhecimento encontrado. No caso de uma empresa, por exemplo, com a quantidade de recursos que se economizou utilizando o conhecimento encontrado.

Podemos aplicar esse mesmo entendimento para a área de governo, e mais especificamente na área de prevenção de corrupção. Um conhecimento útil pode evitar o desperdício de dinheiro público. No caso do combate à corrupção, pode-se identificar culpados e assim, exigir que se reponha aos cofres públicos o dinheiro desviado, por exemplo. Dessa forma, a aplicação de DCBD tem sido utilizada em diversas áreas, tanto no campo da pesquis,a quanto no campo dos negócios e também nas esferas governamentais.

MD é tratada algumas vezes na literatura como sinônimo de DCBD. No entanto, Fayyad et al. (1996) e Han e Kamber (2005) fazem questão de diferenciar os dois conceitos. DCBD é visto como todo o processo de descoberta de conhecimento, passando pelas fases de seleção dos dados, limpeza dos dados, transformação dos dados, MD e terminando na interpretação e avaliação do conhecimento encontrado, como pode ser visto na Figura 2.1.

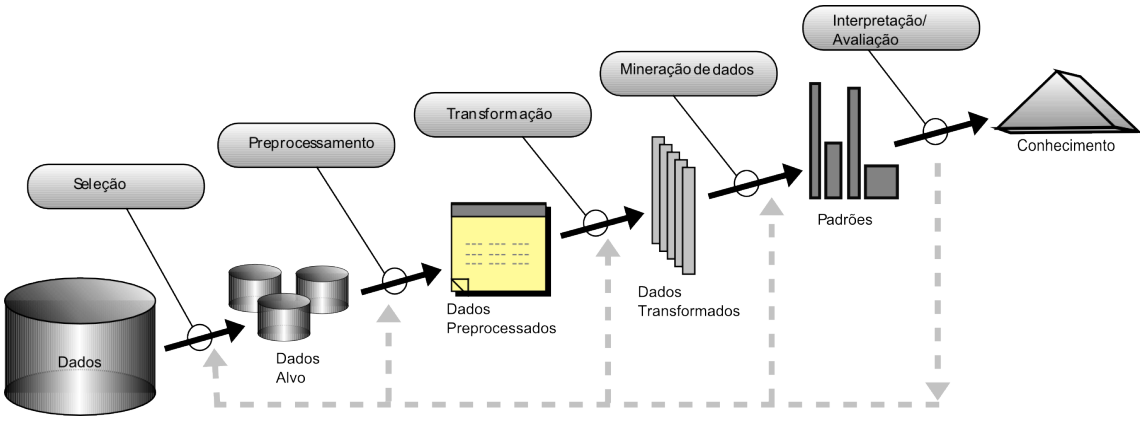


Figura 2.1: Fases do processo de DCBD (Fayyad et al., 1996)

Segundo Han e Kamber (2005), as fases do DCBD podem ser resumidamente conceituadas da seguinte forma:

- Limpeza e seleção dos dados - remoção de impurezas e dados inconsistentes.
- Pré-processamento - onde múltiplos *data sources* podem ser combinados.
- Transformação dos dados - onde os dados são transformados ou consolidados nas formas apropriadas para mineração, utilizando as operações de sumarização e agregação, por exemplo.
- Mineração de dados - uma fase essencial onde métodos inteligentes são aplicados no intuito de extrair padrões (ou modelos) a partir dos dados.
- Avaliação dos padrões - identificar os padrões realmente interessantes baseado em algumas medidas (funções de avaliação).
- Interpretação e apresentação do conhecimento - onde técnicas de visualização e representação do conhecimento são usadas para apresentar o conhecimento minerado ao usuário.

Do ponto de vista de MD e DCBD, a publicação do *CRoss-Industry Standard Process for DM* (CRISP-DM) foi um dos principais marcos no ano de 2000. CRISP-DM é um dos modelos mais usados para desenvolvimento de projetos de MD (Chapman et al., 2000;

Marban et al., 2009). CRISP-DM é uma outra abordagem para a realização do processo de DCBD, como descreveremos na próxima sessão.

2.1.1 CRISP-DM

CRISP-DM é um modelo de referência para o processo de DCBD, mais especificamente para a fase de MD, e tem como objetivo deixar o projeto de MD mais rápido, barato e confiável (Chapman et al., 2000).

CRISP-DM provê uma visão do ciclo de vida de um projeto de MD, dividindo-o em seis fases. Para cada fase, a metodologia apresenta suas respectivas tarefas e as relações entre essas tarefas. Pode-se ver na Figura 2.2 as fases do CRISP-DM. Como a seqüência das fases não é rígida, as setas representam apenas as dependências mais importantes das fases. Por fim, a seta cíclica mais externa da figura, simboliza o refinamento constante das atividades de MD. O fato de encontrar conhecimento não significa que o projeto chegou ao fim, pois o conhecimento adquirido pode ser usado para enriquecer ainda mais os dados e se chegar a outros novos conhecimentos mais significativos e sólidos, inserindo a característica iterativa do processo.

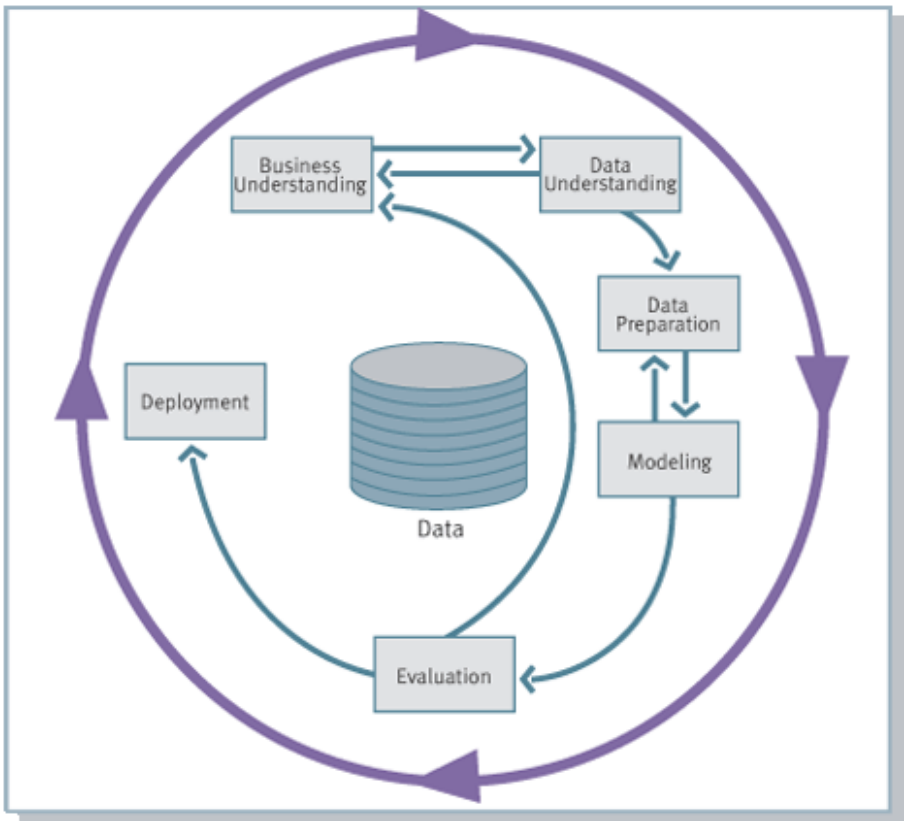


Figura 2.2: Fases do CRISP-DM (Chapman et al., 2000)

Segundo Chapman et al. (2000), as fases do CRISP-DM podem ser descritas resumidamente como se segue:

- Entendimento do Negócio - fase ligada ao entendimento dos objetivos e requisitos do projeto a partir da perspectiva do negócio, ou seja, compreender o domínio da

aplicação. Compreendido este domínio, faz-se necessário também definir o problema no âmbito da MD. Aqui se tem a definição dos objetivos a serem alcançados.

- Entendimento dos Dados - a compreensão dos dados começa com o contato inicial com uma coleção de dados e prossegue com atividades de familiarização com esses dados para identificação de possíveis problemas de qualidade, elaboração de diferentes abordagens para compreensão das informações contidas nos dados. Nesta fase também são feitas elaborações de relatórios da coleta de dados, formato dos dados, quantidade de registros, campos, entre outros. Nesta fase pode-se detectar subconjuntos interessantes capazes de formar hipóteses para detecção de informações escondidas.

- Preparação dos Dados - a construção do *dataset* final a partir dos dados iniciais, sendo que esta fase pode ser executada várias vezes, dada a sua complexidade. Envolve atividades desde a limpeza dos dados até a integração de dados de base e em formatos diferentes. Num contexto mais específico, podem ser definidas nesta fase, estratégias para lidar com dados faltantes, discretização de dados numéricos e até mesmo criação de novos dados a partir de outros existentes.

- Modelagem - relacionada com a escolha das técnicas para solução do problema de MD. Várias técnicas têm requisitos bastante específicos quanto ao formato dos dados de entrada, sendo necessária a interação muitas vezes com a fase de preparação dos dados. Nesta fase é definida também a configuração dos parâmetros dos modelos escolhidos para resolver o problema, e por fim, a aplicação dos mesmos. Podem ser criados nesta fase também planos de testes para futura validação dos modelos escolhidos.

- Avaliação - nesta fase, o modelo construído é analisado no intuito de descobrir se realmente ele alcança os objetivos do negócio. Aqui são revistos os passos executados para construção do modelo observando se existe ainda algum tópico importante do negócio que não foi suficientemente considerado. No final da fase, deve-se decidir sobre o uso dos modelos no negócio ou não.

- Colocação em Uso - a fase de implantação é importante no sentido de que os resultados obtidos precisam ser apresentados num formato que o cliente possa entender. Isso pode ser feito através de um simples relatório, ou mesmo na implantação de um processo mais complexo de repetição do processo de mineração de dados.

2.1.2 Classificação das Técnicas de MD

Segundo Tan et al. (2005), as tarefas de MD são geralmente divididades em duas categorias principais:

- Tarefas Preditivas - tem como objetivo predizer o valor de um atributo particular baseado nos valores de outros atributos. O atributo a ser predito é conhecido como "alvo" ou "variável dependente", enquanto que os atributos usados para fazer a predição são conhecidos como "explanatórios" ou "variáveis independentes".

- Tarefas Descritivas - tem como objetivo derivar padrões (correlações, tendências, grupos, trajetórias e anomalias) que sumarizam as relações subjacentes nos dados.

Tarefas de MD descritivas são frequentemente exploratórias e frequentemente requerem a utilização de técnicas para validar e explanar o resultado (pós-processamento).

Há diversas técnicas de MD, tais como: classificação, clusterização, regras de associação, regras de sequência, regressão, sumarização, entre outras. Destacaremos a seguir duas das principais técnicas utilizadas neste trabalho: clusterização e regras de associação.

2.1.3 Clusterização

Segundo Jain e Dubes (1988), clusterização é uma tarefa descritiva onde se procura identificar um conjunto finito de categorias ou *clusters* para descrever uma informação. Estas categorias podem ser mutuamente exclusivas e exaustivas ou consistir de uma representação mais rica, tal como categorias hierárquicas ou sobrepostas.

A análise de *cluster* está relacionada com outras técnicas que são usadas para dividir objetos de dados em grupos. Por exemplo, a clusterização pode ser considerada como a forma de classificação em que se cria uma rotularização de objetos com rótulos de classe (que são os *clusters*). Entretanto, esses rótulos são derivados unicamente dos dados de forma dinâmica.

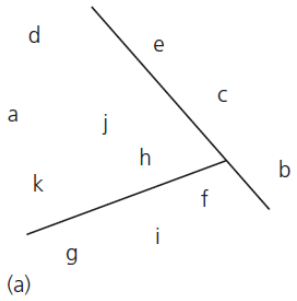
Em contraste, o processo propriamente dito de classificação é uma classificação supervisionada, isto é, objetos novos e não rotulados recebem um rótulo de classe usando um modelo desenvolvido a partir de objetos com rótulos de classes já conhecidos. Por esta razão, análise de *clusters* é, algumas vezes, referida como uma espécie de classificação não supervisionada (Tan et al., 2005).

Cheeseman e Stutz (1996), apresenta como exemplos de aplicação de clusterização no contexto de DCBD: a descoberta de subpopulações homogêneas de consumidores em base de dados de *marketing* e a identificação de subcategorias de espectros a partir de medidas de radiações infravermelhas.

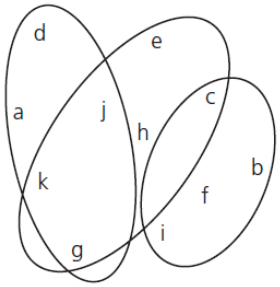
Na Figura 2.3, pode-se observar as diferentes formas de representação de *clusters*. Na Figura 2.3(a), três *clusters* são gerados a partir da repartição do espaço bidimensional. A Figura 2.3(b) mostra um modelo com *clusters* sobrepostos. Alguns algoritmos permitem uma representação mais rica, permitindo que os elementos pertençam a mais de um *cluster*. Na Figura 2.3(c), é mostrada uma representação com a probabilidade de cada elemento pertencer a cada *cluster* do modelo. O algoritmo *Expectation Maximization* (*EM*) utiliza este tipo de respresentação. Outros algoritmos podem produzir uma estrutura hierárquica de *clusters*, como mostrado na Figura 2.3(d).

Segundo Jain e Dubes (1988), o padrão típico de uma atividade de clusterização envolve os seguintes passos:

1. representação padrão - opcionalmente incluindo extração de funcionalidade e/ou seleção;
2. definição de uma medida de proximidade apropriada ao domínio dos dados;
3. clusterização - agrupamento;
4. abstração de dados (se necessário) e
5. avaliação do resultado (se necessário).



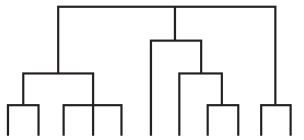
(a)



(b)

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

(c)



(d)

Figura 2.3: Diferentes formas de representação de *clusters* (Witten e Frank, 2005)

Há diversos algoritmos para realização de tarefas de clusterização. Entre eles, destacam-se *K-Means* e *EM*. Esses algoritmos foram apontandos em Wu et al. (2007) como dois dos dez mais influentes algoritmos usados na comunidade científica de MD.

K-Means

K-Means é um algoritmo muito simples de ser implementado e tem complexidade $O(n)$, sendo n o número de elementos a se classificar. Ele opera basicamente escolhendo inicialmente uma partição randômica e prossegue associando os elementos aos *clusters*, baseado na similaridade entre o elemento e o centro do *clusters*.

Jain et al. (1999) resume a execução de *K-Means* em quatro passos:

1. Escolhe-se k centros de *clusters* coincidindo com k elementos escolhidos randomicamente a partir do conjunto inicial;
2. Associa-se cada elemento ao *cluster* com centro mais perto;
3. Recomputam-se os centros dos *clusters*, baseado nos seus membros;
4. Se um critério de convergência não for encontrado, volte ao passo 2. Critérios de convergência típicos: nenhuma (ou mínima) reassociação de elementos a novos centros de *clusters* ou diminuição mínima no erro calculado. Em suma, repita o algoritmo a partir do passo dois, caso os centros dos *clusters* tenham se deslocado.

Segundo Tan et al. (2005), no passo 2, normalmente é utilizado o cálculo da distância euclidiana. No passo 3, os centros podem ser recomputados, conforme a Equação 2.1.

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{x \in C_i} \mathbf{x} \tag{2.1}$$

Segundo Witten e Frank (2005), o critério de convergência adotado pode ser o Critério do Erro Quadrático mostrado na Equação 2.2.

$$E = \sum_{i=1}^{k} \sum_{\mathbf{p} \in C_i} \|\mathbf{p} - \mathbf{m_i}\|^2 \tag{2.2}$$

Onde E é a soma do erro quadrado para todos os objetos no conjunto de dados; p é o ponto no espaço representando um dado objeto; e m_i é a média do *cluster* C_i, sendo p e m_i multidimensionais.

Expectation Maximization

EM também é muito usado para tarefas de clusterização. Segundo Han e Kamber (2005), *EM* é um algoritmo de refinamento iterativo que pode ser usado para encontrar estimativas de parâmetro. Pode ser visto como uma extensão do paradigma *k-means*, que associa um objeto ao *cluster* que lhe é mais similar, baseado na média encontrada. No entanto, ao invés de associar cada objeto a um único *cluster*, *EM* pode associar objetos a mais de um *cluster*, definindo um peso a cada associação, representando a probabilidade daquele objeto pertencer ao *cluster*. O algoritmo *EM* computa as probabilidades dos membros dos *clusters* baseado em uma ou mais distribuições de probabilidade. A meta do algoritmo é maximizar a verossimilhança (*likelihood*) dos dados nos *clusters* finais.

Para compreender o funcionamento de *EM*, introduziremos um exemplo bastante intuitivo apresentado em Do e Batzoglou (2008). Suponha duas moedas "A" e "B", num jogo de lançamentos de moedas. Temos que θ_A é a probabilidade da moeda A dar "cara" (H), e $(1 - \theta_A)$, a probabilidade da moeda A dar "coroa" (T). De forma similtar temos também θ_B. Nossa meta é estimar $\theta = (\theta_A, \theta_B)$, repetindo o seguinte procedimento cinco vezes: randomicamente, escolheremos uma moeda (com igual probabilidade) e executaremos 10 lançamentos com a moeda escolhida. Sendo assim, serão 50 lançamentos de moeda, conforme Figura 2.4.

Durante nosso experimento, suponha que sejam mantidas trilhas de dois vetores $x = \{x_1, x_2, ..., x_5\}$ e $z = \{z_1, z_2, ..., z_5\}$, onde $x_i \in \{0, 1, ..., 10\}$ é o número de caras observado durante o i-ésimo conjunto de lançamento da moeda, e $z_i \in \{A, B\}$ é o identificador da moeda usada durante o i-ésimo conjunto de lançamentos.

Uma simples forma de calcular θ_A e θ_B é retornar as proporções de cara (H) para cada moeda:

$$\begin{aligned} \hat{\theta_A} &= \frac{\text{número de caras obtidos com moeda A}}{\text{total de caras e coroas obtidos com a moeda A}} \\ \hat{\theta_B} &= \frac{\text{número de caras obtidos com moeda B}}{\text{total de caras e coroas obtidos com a moeda B}} \end{aligned} \tag{2.3}$$

As Equações 2.3, apresentam de forma intuitiva, o fato conhecido como Máxima Verossimilhança. De forma simples, esse método mede a qualidade de um modelo estatístico baseado na probabilidade de se associar os dados observados. O exemplo pode ser conferido na Figura 2.4.

Agora, tomemos um exemplo mais complexo a partir do que apresentamos. Imagine o problema de estimativa, onde são dados os conjuntos de "cara" obtidos (x), mas sem as identidades z de qual moeda foi usada. Nós passamos a referir z como variável escondida,

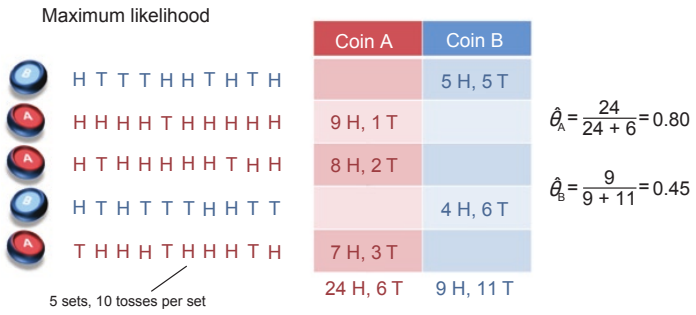


Figura 2.4: Estimativa de Máximo Verossimilhança. Com a soma de cara e coroa, estima-se a probabilidade do resultado ser "cara"(Do e Batzoglou, 2008)

ou *fator latente*. Agora, computar as proporções de "caras" para cada moeda não é mais possível, dado que não sabemos qual moeda se refere a cada conjunto de lançamentos. Mas se, de alguma forma, nós podemos completar os dados (em nosso caso, escolhendo corretamente qual moeda foi usada em cada um dos cinco conjuntos), então nós podemos reduzir a estimativa de parâmetro desse problema com dados incompletos para estimativa de máxima verossimilhança com dados completos.

Podemos montar um esquema para completar os dados faltantes da seguinte forma:

1. Comece com alguns parâmetros iniciais $\hat{\theta}^{(t)} = (\hat{\theta_A}^{(t)}, \hat{\theta_B}^{(t)})$
2. Determine para cada um dos cinco conjuntos qual moeda A ou B foi mais semelhante para ter gerado os lançamentos observados (usando as estimativas de parâmetro atuais).
3. Então assuma que estas complementações (que são as moedas escolhidas e associadas) estão corretas, e aplique a estimativa regular de máxima verossimilhança para encontrar $\hat{\theta}^{(t+1)} = (\hat{\theta_A}^{(t+1)}, \hat{\theta_B}^{(t+1)})$.
4. Repita os dois últimos passos até o valor convergir.

O algoritmo *EM* é um refinamento com essas mesmas idéias. Ao invés de pegar o único complemento mais semelhante da moeda faltante em cada iteração, *EM* computa probabilidades para cada possível complemento de dado faltante, usando os atuais parâmetros $\hat{\theta}^{(t)}$. Essas probabilidades são usadas para criar um conjunto ponderado consistindo de todos os possíveis complementos dos dados.

O exemplo de estimar a moeda certa pode ser acompanhado também na Figura 2.5:

1. EM inicia com um valor inicial dos parâmetros: $\hat{\theta_A}^{(t)} = 0.60$ e $\hat{\theta_B}^{(t)} = 0.50$.
2. No passo-E (*E-step*), uma distribuição de probabilidade sobre possíveis complementos é computada usando os parâmetros atuais. Por exemplo, na primeira sequência, onde são 5 caras e 5 coroas, temos:

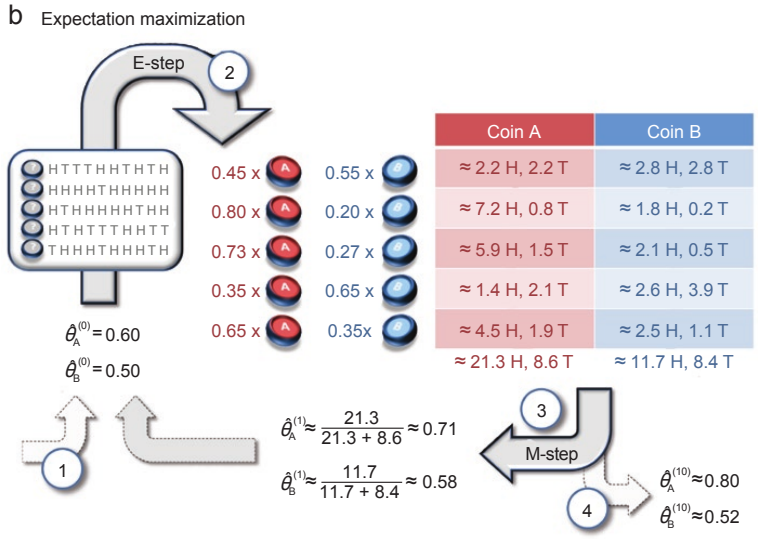


Figura 2.5: Funcionamento do *EM* para o problema da moeda (Do e Batzoglou, 2008)

$$
\begin{array}{lrcl}
\text{Prob. de A ser cara} & \hat{\theta_A}^{(t)} & = & 0.60 \\
\text{Prob. de A ser coroa} & 1 - \hat{\theta_A}^{(t)} & = & 0.40 \\
 & prob(A|x_1, \hat{\theta_A}^{(t)}) & = & \frac{0.60^5 * 0.40^5}{0.60^5 * 0.40^5 + 0.50^5 * 0.50^5} \\
 & & \simeq & 0.45
\end{array}
$$

3. Os números mostrados na tabela da Figura 2.5 são os números esperados de "caras" e "coroas", de acordo com a distribuição. A terceira linha e a primeira coluna da tabela são calculadas da seguinte forma:

$$
\begin{array}{lrcl}
\text{Distribuição} & \text{8 caras e 2 coroas} & & \\
\text{Prob. de ser moeda A} & 0.73 & & \\
\text{Prob. de ser moeda B} & 0.27 & & \\
\text{Moeda A} & = 0.73 * 8 & = & 5.84 \simeq 5.9 \text{ caras} \\
\text{Moeda A} & = 0.73 * 2 & = & 1.46 \simeq 1.5 \text{ coroas}
\end{array}
$$

4. No passo-M (M-step), novos parâmetros são determinados usando os complementos atuais.

5. Depois de várias repetições, o algoritmo converge ($\hat{\theta_A}^{(10)} = 0.80$ e $\hat{\theta_B}^{(10)} = 0.52$, valores próximos dos originais $\hat{\theta_A} = 0.80$ e $\hat{\theta_B} = 0.45$).

Segundo Do e Batzoglou (2008), EM alterna entre dois passos: (i) escolher uma distribuição probabilística para completar os dados faltantes a partir do atual modelo (conhecido como *E-step, Expectation*); (ii) reestimar os parâmetros do modelo usando os dados estimados (conhecido como *M-step*, Maximization). Para discussões mais elaboradas sobre EM, ver Dempster et al. (1977), Borman (2004), Prescher (2004), Wu et al. (2007).

2.1.4 Regras de Associação

Técnica de MD que consiste em descobrir relações fortes entre determinados atributos. Essa técnica tem a capacidade de detectar padrões em forma de regras que associam valores de atributos num determinado conjunto de dados. Essas regras são expressas em forma de conjunções do tipo $atrib_1 = valor_1, atrib_2 = valor_2 \ \ ..., atrib_m = valor_m \rightarrow atrib_{m+1} = valor_{m+1}, atrib_{m+2} = valor_{m+2} \ \ ..., atrib_n$, onde *atrib* é um atributo do conjunto de dados e *valor* é o valor do atributo identificado na regra.

A cobertura das regras de associação é medida pela probabilidade da regra se repetir no conjunto de dados e é chamada também de *suporte*. A acurácia da regra, chamada de confiança, é o percentual de instâncias preditas corretamente pela regra. A confiança pode ser usada para medir a qualidade da regra, ressaltando a corretude da inferência da mesma. Uma regra com alta confiança significa que a inferência funciona bem no universo aonde a regra foi definida. Assim, quanto mais alto o suporte e a confiança, mais forte e de melhor qualidade é a Regra de Associação.

Uma definição formal da técnica pode ser encontrada em Han e Kamber (2005):

Definição 1 *Seja $I = \{I_1, I_2, ..., I_M\}$ um conjunto de itens e D os dados da base, onde T é o conjunto de transações de D, tal que $T \subseteq I$. Sejam também A e B conjuntos de itens. Uma regra de associação é uma implicação da forma $A \Rightarrow B$, onde $A \subset I$, $B \subset I$, e $A \cap B = \phi$. A regra $A \Rightarrow B$ se aplica no conjunto de transações D com suporte s, onde s é o percentual de transações em D, que contém $A \cup B$, isto é, a probabilidade $P(A \cup B)$. A regra $A \Rightarrow B$ tem confiança c no conjunto de transações D, onde c é o percentual de transações em D contendo A, que também contém B, isto é, a probabilidade condicional $P(B|A)$.*

Um exemplo de regra de associação extraída da base apresentada na Tabela 2.1 é:

$$temperatura = frio, umidade = normal \rightarrow jogar = sim$$

Neste regra, estão associados três atributos: temperatura, umidade e jogar. O lado esquerdo da regra ($temperatura = frio, umidade = normal$) está presente em quatro instâncias da Tabela 2.1, respectivamente nas linhas 5, 6, 7 e 9. O suporte dessa regra é quatro, ou 29%. Dessas quatro linhas, em apenas três o lado direito da regra aparece, linhas 5, 7 e 9. Desta forma, a confiança da regra é calculada $3/4 = 0,75$, ou seja 75%.

Segundo Tan et al. (2005), é proibitivo o cálculo de todas as regras de associação, posto que a abordagem de força-bruta neste caso é exponencial, na quantidade de itens associativos analisados. A complexidade de espaço para cálculo das regras por força bruta tem a seguinte função: $R = 3^d - 2^d + 1$, onde d é o número de itens.

Segundo Witten e Frank (2005), a diferença entre classificação e regras de associação é que estas podem prever padrões com qualquer atributo, e não só da classe selecionada. Diferentes regras de associação expressam diferentes regularidades subjacentes no conjunto de dados, cada uma predizendo coisas diferentes.

Algoritmo Apriori

Apriori é um algoritmo seminal proposto em Agrawal e Srikant (1994), para mineração de conjunto de itens frequentes para regras de associação.

Tabela 2.1: Base de dados com informações sobre o tempo e decisão sobre sair para jogar ou não

	Tempo	**Temperatura**	**Umidade**	**Ventando**	**Jogar?**
1	ensolarado	quente	alta	falso	não
2	ensolarado	quente	alta	verdadeiro	não
3	nuvens	quente	alta	falso	sim
4	chuvoso	moderado	alta	falso	sim
5	chuvoso	frio	normal	falso	sim
6	chuvoso	frio	normal	verdadeiro	não
7	nuvens	frio	normal	verdadeiro	sim
8	ensolarado	moderado	alta	falso	não
9	ensolarado	frio	normal	falso	sim
10	chuvoso	moderado	normal	falso	não
11	ensolarado	moderado	normal	verdadeiro	sim
12	nuvens	moderado	alta	verdadeiro	sim
13	nuvens	quente	normal	falso	sim
14	chuvoso	moderado	alta	verdadeiro	não

Apriori emprega uma abordagem iterativa conhecida como *level-wise search*, para geração de regras de associação, onde cada nível corresponde ao número de itens que pertencem ao consequente da regra. Inicialmente, todas as regras de confiança alta que tem apenas um item no consequente da regra são extraídas. Estas regras são então usadas para gerar novas regras candidatas. Por exemplo, se $\{a, c, d\} \rightarrow \{b\}$ e $\{a, b, d\} \rightarrow \{c\}$ são regras de confiança alta, então a regra candidata $\{a, d\} \rightarrow \{b, c\}$ é gerada através da fusão das duas outras. Analogamente, suponha que a confiança da regra $\{b, c, d\} \rightarrow \{a\}$ é baixa. Então todas as regras contendo o item a no seu consequente pode ser descartada, por exemplo $\{c, d\} \rightarrow \{a, b\}$, $\{b, d\} \rightarrow \{a, c\}$. Este comportamento é formalizado em Tan et al. (2005) da seguinte forma:

Teorema 1 *(Poda Baseada em Confiança) Se numa regra $X \rightarrow Y$, X não satisfaz o limite mínimo de confiança, então em qualquer regra $X' \rightarrow Y$, X', sendo um subconjunto de X, não satisfará o limite mínimo de confiança também.*

Outra estratégia usada para se contornar o problema de mapeamento de todas as regras de associação, e reduzir o número de conjuntos candidatos a se tornarem regras de associação, é poda baseada no suporte. Para isso, os algoritmos de associação podem trabalhar também sob esse princípio (Tan et al., 2005):

Teorema 2 *(Princípio Apriori - baseado no suporte) Se um conjunto é frequente, então todos os seus subconjuntos também serão frequentes.*

Ou seja, se temos um conjunto de 3 atributos $\{a, b, c\}$ que frequentemente aparece nos registros de dados, logo seus subconjuntos $\{a, b\}$, $\{a, c\}$, $\{b, c\}$, $\{a\}$, $\{b\}$, $\{c\}$ também serão, no mínimo, tão frequentes quanto.

O corolário do teorema é que se um subconjunto não é frequente, os seus superconjuntos também não serão. Por exemplo, se $\{a, b\}$ não é freqüente, $\{a, b, c\}$, $\{a, b, d\}$, ..., $\{a, b, ...\}$, também não serão.

Baseando-se nesse princípio, quando se define um valor de suporte (frequência mínima do conjunto), todos os conjuntos que não alcançam a frequência definida no suporte são descartados, reduzindo consideravelmente a complexidade de tempo e espaço no momento de execução do algoritmo.

Na próxima seção serão apresentados os conceitos relacionados a SMA para possibilitar a proposta na área de AMII.

2.2 Sistemas Multiagentes

A definição intuitiva de SMA é um sistema computacional composto por vários agentes de software que interagem entre si em um ambiente distribuído. Esse ramo da Inteligência Artificial (IA) começou a ser estudado na segunda metade da década de 70, sendo denominado Inteligência Artificial Distribuída (IAD) (Weiss, 1999).

SMA é hoje um campo de pesquisa da IAD, com aplicação de conceitos e idéias de muitas outras disciplinas, incluindo IA, Ciência da Computação, Sociologia, Economia, Administração e Filosofia. Segundo Weiss (1999), IAD é o estudo, construção, e aplicação de sistemas multiagentes, isto é, sistemas em que diversos agentes inteligentes interagem procurando atingir seus objetivos ou para realizar algumas tarefas definidas.

Segundo Russell e Norvig (2010), SMA é um sistema onde muitos agentes interagem indiretamente, isto é, agindo no ambiente, ou mesmo diretamente através de comunicação ou negociação. Os agentes podem decidir cooperar por benefícios mútuos ou podem competir para atender seus próprios interesses.

2.2.1 Agentes

Para entendermos melhor um SMA passamos a detalhar as características de um agente. Segundo Wooldridge (2002), um agente é um sistema de computação que é inserido em um ambiente e que é capaz de ter ações autônomas nesse ambiente no intuito de alcançar seus objetivos projetados. Em outras palavras, o agente tem a capacidade de perceber um ambiente e agir autonomamente, alterando ou não este ambiente, para alcançar suas metas. Segundo o autor, são características de um agente: autonomia, reação, interação e iniciativa. A Figura 2.6 mostra um esboço da arquitetura de alto nível de um agente. Esta figura ilustra bem a definição de agente encontrada em Russell e Norvig (2010), que diz que um agente é tudo que pode ser visto, percebendo seu ambiente através de sensores e agindo sobre ele através de seus atuadores.

Alguns autores diferenciam agentes de agentes inteligentes. Wooldridge e Jennings (1995) citam capacidades especiais dos agentes inteligentes, além da autonomia, a saber:

- Reação- agentes inteligentes são capazes de perceber seus ambientes e responder às mudanças que ocorrem neles em tempo hábil, no intuito de satisfazer os seus objetivos projetados.
- Proatividade- agentes inteligentes são hábeis para tomar iniciativas, no intuito de satisfazer os seus objetivos projetados.
- Capacidade social- agentes inteligentes são capazes de interagir com outros agentes (e possivelmente humanos), no intuito de satisfazer os objetivos projetados.

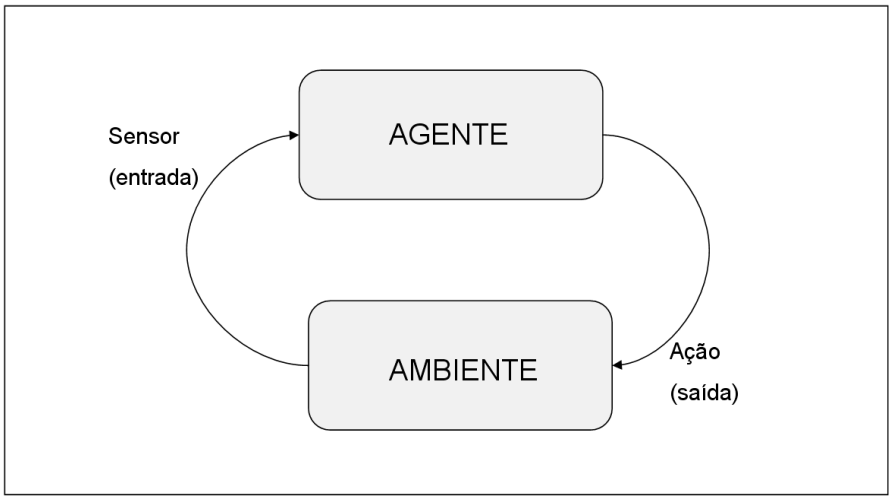


Figura 2.6: Um agente no seu ambiente. Adaptado de Russell e Norvig (2010)

Segundo Russell e Norvig (2010), agentes podem ser classificados em quatro tipos básicos:

- Agente reativo - decide o que fazer sem referenciar um histórico. O agente apenas escolhe uma ação baseada na sua percepção. É a configuração mais simples de um agente, no entanto, sua inteligência se torna limitada.
- Agente reativo com estado - é um agente reativo mais complexo, pois mantém um estado interno que depende de seu histórico de percepções e ações. Esse agente é também chamado de agente baseado em modelo. A cada nova percepção, o agente atualiza seu estado interno, e suas ações agora dependem não só da percepção recebida, mas também de seu estado interno.
- Agente baseado em objetivo - é um agente mais complexo, que mantém registro interno do estado do mundo baseado em suas percepções e também um conjunto de metas que ele tenta alcançar. Desta forma, o agente escolhe sua ação que irá eventualmente o ajudar a alcançar seus objetivos, levando em conta sua análise do estado atual do mundo.
- Agente baseado em utilidade - este agente não só tem objetivos, mas ele é capaz de mensurar o grau de satisfação a partir do estado em que se encontra. Isso é feito através da função de utilidade que o agente mantém internamente. É extremamente útil em casos onde o agente possui objetivos conflitantes. Por exemplo, no caso de um agente motorista de taxi, são metas conflitantes do agente dirigir rápido e de forma segura. Através da função de utilidade, o agente consegue equilibrar esses dois objetivos mensurando o quão rápido ele pode ir mantendo a segurança no percurso.

Ambiente

Segundo Russell e Norvig (2010), o ambiente onde o agente atua é onde o problema se encontra, para o qual o agente é a solução. Assim, os autores especificam várias propriedades que podem ser usadas para classificar os ambientes.

- Completamente Observável vs. Parcialmente Observável
  Um ambiente é completamente observável se os sensores do agentes detectam todos os aspectos que são relevantes para a escolha da ação. Um ambiente pode ser parcialmente observável por causa dos ruídos ou sensores não apurados, ou simplesmente porque nem todos os estados do ambiente podem ser percebidos pelos sensores do agente.

- Determinístico vs. Estocástico
  Se o próximo estado do ambiente é completamente determinado pelo atual estado e a ação executada pelo agente, então dizemos que o ambiente é determinístico, caso contrário, estocástico. Um tabuleiro de xadrez, por exemplo, onde cada agente movimenta sua respectiva peça, sendo que as movimentações subsequentes são determinadas pelo estado corrente do tabuleiro, o respectivo movimento do agente e as regras do jogo, é considerado um ambiente determinístico. Já no cenário de um agente motorista de taxi no trânsito, o ambiente é considerado estocástico, pois não é previsível.

- Episódico vs. Sequencial
  Num ambiente episódico, a experiência do agente é dividida em episódios atômicos. Cada episódio consiste do agente percebendo e então executando uma única ação. O próximo episódio não depende das ações tomadas nos episódios anteriores. No ambiente episódico, a escolha da ação em cada episódio depende unicamente do próprio episódio. Por exemplo, um agente que necessita localizar partes defeituosas numa linha de montagem, baseia cada decisão na parte atual analisada, sem considerar decisões tomadas anteriormente. Além disso, decisão atual não irá alterar o fato da próxima parte analisada apresentar ou não um defeito.

- Estático vs. Dinâmico
  Se o ambiente pode mudar enquanto um agente está deliberando, então dizemos que o ambiente é dinâmico para o agente; caso contrário, ele é estático. Ambientes estáticos são fáceis de manusear porque o agente não necessita monitorar o ambiente enquanto ele está decidindo uma ação. Ambientes dinâmicos, por outro lado, estão sempre se modificando, com o decorrer do tempo.

- Discreto vs. Contínuo
  A distinção entre ambiente discreto e contínuo pode ser aplicada ao estado do ambiente, considerando o tempo, as percepções e ações do agente. Por exemplo, todas as combinações de um jogo de xadrez podem ser previstas, ou seja, há um número discreto de estados. Já todas as ações possíveis de um motorista de taxi no trânsito não podem ser previstas, sendo tipicamente um ambiente contínuo.

- Agente Único vs. Multiagente
  Um agente resolvendo palavras cruzadas sozinho está nitidamente num ambiente com agente único, enquanto que um agente jogando xadrez está em um ambiente com mais de um agente. Essa distinção de ambientes faz diferença na visão do agente, já que em um ambiente multiagente ele precisa considerar a presença dos outros agentes para maximizar sua medida de desempenho. Em um ambiente multiagentes competitivo, para que um agente maximize seu desempenho, ele precisa minimizar a medida de performance dos agentes concorrentes. Isso acontece tam-

Tabela 2.2: Exemplos de tipos de agentes e os respectivos PEAS (traduzido de Russell e Norvig (2010))

Tipo de Agente	**Medida Desempenho (P)**	**Ambiente (E)**	**Atuadores (A)**	**Sensores (S)**
Sistema de diagnóstico médico	Saúde do paciente, custos minimizados, processos	Paciente, hospital	Testes, diagnósticos, tratamentos, encaminhamentos	Entrada de teclado de sintomas, respostas, resposta do paciente
Tutor de inglês interativo	Maximizar o ponto do estudante no teste	Grupo de estudantes, agencia de testes	Display de exercícios, sugestões, correções	Entrada de teclado
Motorista de Taxi	Viagem segura, rápida, confortável, e maximizar lucros	Estradas, tráfeto, pedestres, clientes	acelerador, freio, sinal, buzina, volante, display	Câmeras, sonar, velocímetro, GPS, acelerômetro, sensores de motor, teclado

bém em um ambiente multiagente cooperativo, onde o agente precisará se preocupar com o desempenho dos demais agentes para que todos possam atingir o objetivo em conjunto. Os problemas nos ambientes de agente único e multiagentes são muitas vezes completamente diferentes. A comunicação, por exemplo, frequentemente emerge como um comportamento racional em sistemas multiagentes. Em ambientes competitivos, o comportamento estocástico é racional, porque evita que sejam criadas armadilhas explorando a previsibilidade. As características do agente e do ambiente devem ser consideradas no projeto de agentes em um SMA.

2.2.2 Projeto de um Agente

Há diversos tipos de agentes definidos na literatura. Wooldridge (2002) fala de agentes puramente reativos, agentes com estados, agentes orientados a metas, entre outros. Já Russell e Norvig (2010), conforme exposto na Seção 2.2.1, enumera quatro tipos distintos: reflexivos (reativos), reflexivos baseado em modelo (reativo com estado), orientado a metas (baseado em objetivos) e orientado a utilidade (baseado em utilidade).

Uma proposta bastante aceita para projeto de agentes é feita em Russell e Norvig (2010). O Autor sugere, para projeto de agentes, definir a medida de desempenho, o ambiente, os atuadores e sensores do agente. Essa proposta recebeu o acrônimo de PEAS (*Performance, Environment, Actuators, Sensors*). Inicialmente, em Russell e Norvig (1995), os autores fixaram o acrônimo PAGE (*Percepts, Actions, Goals, Environment*); no entanto, as metas estavam muito mais relacionadas à medida de desempenho do agente, sendo substituído nas edições subsequentes pelo acrônimo PEAS. Um exemplo de definição do PEAS em projeto de agentes pode ser visto na Tabela 2.2.

Agente de Mineração de Dados

Um Agente de Mineração de Dados, ou simplesmente um Agente de Mineração, pode ser conceituado como um agente com o propósito de encontrar conhecimento de forma eficiente, através da utilização de algoritmos de MD.

Moemeng et al. (2009) discute a utilização de agentes nas tarefas de MD distribuída. SMA apresenta, por sua natureza, a característica de descentralização, o que parece se encaixar muito bem com os requisitos de MD distribuída. Em cada base de dados, ou mesmo *datasets*, estratégias de mineração podem ser aplicadas especificamente num determinado domínio de dados, juntando posteriormente os conhecimentos encontrados de forma consolidada. E toda essa tarefa pode ser facilitada com a utilização de agentes específicos de mineração.

O ambiente de um agente de mineração é normalmente composto pelas bases de dados que ele tem acesso. Pode-se adicionar a esse ambiente, bases de conhecimento compartilhadas por outros agentes de mineração, adquiridos na maioria das vezes através da execução de algoritmos de MD.

A medida de desempenho é a qualidade do conhecimento encontrado. No entanto, segundo Fayyad et al. (1996), a compreensibilidade dos padrões encontrados pode ser subjetiva, necessitando da definição de parâmetros de mensuração dos mesmos, podendo-se utilizar, por exemplo, funções já existentes para medidas de frequência, acurácia, similaridade, a fim de auxiliar na mensuração da qualidade (Tan et al., 2005).

2.2.3 Características de SMA

Em Huhns e Stephens (1999), encontra-se enumeradas três características principais de ambientes multiagentes:

1. Ambientes multiagentes fornecem uma infraestrutura que especifica protocolos de comunicação e interação.
2. SMA são tipicamente abertos e não têm necessariamente centralização.
3. SMA contém agentes que são autônomos e distribuídos, e podem ser cooperativos, ou individualista (*self-interested*).

A própria existência de mais de um agente com características autônomas traz a necessidade da existência de meios de comunicação e interação entre eles. O comportamento de cooperação, por exemplo, é facilitado na troca de informações. Da mesma forma, a existência de regras de interação entre agentes num jogo aonde há competição pode ser usada pelos agentes na criação de estratégias para maximização de seus desempenhos.

Protocolo de Comunicação

Comunicação é um tópico de central importância na Ciência da Computação, especialmente quando se trata de problemas de sistemas concorrentes e a sincronização de múltiplos processos. Da mesma forma, em ambiente distribuído, como o de um SMA, a comunicação é de essencial importância.

Segundo Wooldridge (2002), o estudo das linguagens de comunicação entre agentes, a *Speech Act Teory*, que trata a comunicação como uma ação, informou e influenciou

Tabela 2.3: Parâmetros de mensagem no FIPA-ACL

Parâmetro	**Categoria**
performative	Tipo do ato de comunicação
sender	Participante na comunicação
receiver	Participante na comunicação
reply-to	Participante na comunicação
content	Conteúdo da mensagem
language	Descrição do Conteúdo
encoding	Descrição do Conteúdo
ontology	Descrição do Conteúdo
protocol	Controle da conversa
conversation-id	Controle da conversa
reply-with	Controle da conversa
in-reply-to	Controle da conversa
reply-by	Controle da conversa

diretamente algumas das linguagens que foram desenvolvidas especificamente para comunicação de agentes. Segundo o autor, na início da década de 90, foi formado o *Knowledge Sharing Effort* (KSE) com a missão de desenvolver protocolos para troca de conhecimento representado entre sistema de informações autônomos. A partir dessa idéia, KSE gerou dois protocolos: *Knowledge Interchange Format* (KIF) e *Knowledge Query and Manipulation Language* (KQML), este último, tornando-se seminal na elaboração do *FIPA Agent Communication Language* (FIPA-ACL).

The Foundation for Intelligent Physical Agents (FIPA) é uma organização internacional que se dedica a promover a indústria de agentes inteligentes através do desenvolvimento aberto de especificações, apoiando a interoperabilidade entre agentes e aplicações baseada em agentes (FIPA, 2002a). Os parâmetros de mensagem definidos no FIPA-ACL podem ser vistos na Tabela 2.3. De acordo com a especificação, apenas o parâmetro *performative* é mandatório. No entanto, é esperado da maioria das mensagens, os parâmetros *sender*, *receiver*, e *content*.

Na especificação definida em FIPA (2002b), são encontradas 22 performativas, ou tipos de atos de comunicação. Wooldridge (2002) classifica esses tipos em cinco categorias diferentes enumeradas a seguir:

1. Requisição de Informação - *query-if, query-ref, cancel, subscribe*;
2. Passagem de Informação - *confirm, desconfirm, inform, inform-if, inform-ref*;
3. Negociação - *propose, accept-proposal, reject-proposal, cfp*;
4. Execução de Ações - *request, request-when, request-whenever, refuse, proxy, propagate, agree, cancel*;
5. Tratamento de Erro - *failure, not-understood*;

Um exemplo de troca de mensagens utilizando FIPA-ACL entre dois agentes, A e B, pode ser visto a seguir:

1. Agente i solicita ao agente j que submeta sua proposta para vender 50 ameixas (*plum*), utilizando a performativa *cfp (call for proposal)*. Nessa proposta, é dada a condição de que o preço deve ser menor que $10.

```
(cfp
  :sender (agent-identifier :name j)
  :receiver (set (agent-identifier :name i))
  :content
    "((action (agent-identifier :name i)
      (sell plum 50))
     (any ?x (and (= (price plum) ?x) (< ?x 10))))"
  :ontology fruit-market
  :language fipa-sl)
```

2. O agente B, então, propõe ao agente A vender 50 ameixas por $5:

```
(propose
  :sender (agent-identifier :name j)
  :receiver (set (agent-identifier :name i))
  :content
    "((action j (sell plum 50))
     (= (any ?x (and (= (price plum) ?x) (< ?x 10))) 5)"
  :ontology fruit-market
  :in-reply-to proposal2
  :language fipa-sl)
```

No exemplo mostrado anteriormente, é usada a ontologia fictícia *fruit-market*. Segundo Hendler (1999), ontologia é uma definição formal de um corpo de conhecimento, a qual é adotada quando a comunicação envolve domínios específicos gerando a necessidade de definição formal de termos de maior complexidade na comunicação.

Ontologia

Segundo definição encontrada em Wooldridge (2002), ontologia é uma definição formal de um corpo de conhecimento. O mais típico tipo de ontologia usado na construção de agentes envolve um componente estrutural. Essencialmente, é uma taxonomia de relações de classe e subclasse acopladas com definições de relações entre elas. Huhns e Stephens (1999) define ontologia como a especificação de objetos, conceitos e relações numa área de interesse.

O importante na utilização de uma ontologia é definir uma estrutura comum de significados para ser usados pelos agentes na comunicação sem o risco de haver ambigüidades na comunicação. A discussão sobre ontologia remete à necessidade do agente representar seu conhecimento no vocabulário específico da ontologia utilizada (Huhns e Stephens, 1999).

Um dos parâmetros de mensagem do FIPA-ACL é a definição da ontologia usada na comunicação. O modelo de comunicação FIPA-ACL assume que quando os agentes se comunicam, eles compartilham uma ontologia de comunicação (Figura 2.7).

Figura 2.7: Modelo de comunicação baseado em ontologia (adaptado de FIPA (2001))

Num ambiente aberto, agentes podem ser projetados sob várias ontologias, algumas implícitas e outras explícitas. As ontologias explícitas são as consideradas declarativamente representadas, ao contrário das implícitas que são codificadas através de procedimentos do sistema. No caso das ontologias explícitas, os agentes precisam compartilhar intrinsecamente a mesma ontologia para ser capaz de se comunicar, e isso é uma restrição forte em ambientes abertos, projetados por diferentes programadores (FIPA, 2001).

Desta forma, para se ter uma comunicação efetiva, os agentes devem também compartilhar uma ontologia que seja compatível com o domínio da aplicação compartilhada. Num SMA envolvendo técnicas de MD, a ontologia utilizada deverá conter conceitos relativos às estruturas de conhecimento procurado dos modelos de MD, suas relações, as ações realizadas por agentes no SMA, entre outros.

Noy e McGuinness (2001) propõe o que chama de uma Metodologia Simples de Engenharia de Conhecimento, para desenvolvimento de ontologias, conhecido como Método 101. A proposta é interativa, iniciando-se com um rascunho simples do que seria a ontologia, e depois revisando e refinando sucessivas vezes até chegar no modelo desejado. São propostos sete passos para se desenvolver uma ontologia, a saber:

1. Determinar o domínio e o escopo da ontologia - neste passo, normalmente são feitas as maiores revisões, pois se trata de um dos pontos mais importantes da definição da ontologia. Aqui são respondidas questões tais como qual domínio a ontologia cobrirá, para que se está usando esta ontologia, para quais tipos de questões a informação na ontologia deverá prover respostas e quem usará esta ontologia;

2. Considere reusar ontologias existentes - o reuso de existentes ontologias pode ser um requisito se o sistema em questão necessita interagir com outras aplicações que já tem se comprometido com uma ontologia particular ou vocabulários controlados. Muitas ontologias já são disponíveis eletronicamente e podem normalmente ser importadas para o ambiente de desenvolvimento de ontologia que se está usando;

3. Enumerar os termos importantes na ontologia - é útil listar todos os termos que estão relacionados com o tema da ontologia que se está desenvolvendo. Neste passo, não se leva em conta as relações hierárquicas entre esses termos, apenas uma listagem geral dos termos;

4. Definir as classes e a hierarquia de classes - um processo de desenvolvimento *top-down* inicia com a definição do conceito mais geral no domínio e subsequente especialização dos conceitos. Por exemplo, se estamos falando de *Vinho*, a classe vinho de-

verá ser especializada posteriormente em outras subclasses tais como *Branco*, *Rosé*, *Tinto*. A sub-classe *Tinto* ainda poderá ser especializada em outras sub-classes tais como *Cabernet Sauvignon*, *Merlot*, entre outros. Assim, neste passo, os conceitos listados no passo anterior são postos na hierarquia de conceitos, fazendo com que a ontologia já comece a tomar forma, através de uma taxonomia dos conceitos;

5. Definir as propriedades das classes (*slots*) - classes sozinhas não podem prover as informações necessárias que uma ontologia deve responder. Desta forma, uma vez definida as classes, neste passo, deve se definir a estrutura interna dos conceitos. Deve-se lembrar também que as propriedades definidas num conceito são herdadas pelas suas subclasses. As propriedades poderão ser de tipos primitivos, ou mesmo um conceito já definido na taxonomia. No caso do *Vinho*, poderíamos ter as propriedades *cor*, *corpo*, *local*. A propriedade *local* seria da classe *Vinicultura*;

6. Definir as restrições nas propriedades - as propriedades podem ter diferentes restrições, tais como tipo, valor permitido, cardinalidade, e outras características de valores que a propriedade carrega. O tipo do valor da propriedade pode ser um tipo primitivo ou uma instância. A cardinalidade pode ser um para um, um para muitos, entre outros. Neste passo que serão definidas essas características mais específicas das propriedades;

7. Criação das instâncias - neste último passo, são criadas as instâncias individuais das classes na hierarquia. Definir uma instância de uma classe requer: (1) escolher a classe, (2) criar uma instância individual da classe, e (3) preencher os valores das propriedades.

Ao completar os sete passos, finaliza-se o ciclo proposto para modelagem de ontologia, segundo o Método 101.

Existem várias ferramentas disponíveis na internet para modelagem de ontologia. Alguns exemplos são *Protégé* (`http://protege.stanford.edu`),*OntoTrack* (`http://www.informatik.uni-ulm.de/ki/ontotrack/`) e *Karlsruhe Ontology Management Infrastructure* (KAON) (`http://kaon.semanticweb.org`). Uma linguagem muito utilizada para a definição de ontologias é a *Web Ontology Language* (OWL), recomendada pelo *World Wide Web Consortium* (W3C), na plataforma da web semântica.

Adotamos neste trabalho o *Protégé*, pelo fato desta ferramenta já ser bem documentada muito difundida em outras pesquisas. Além disso, *Protégé* disponibiliza o *plugin Ontology-BeanGenerator* (`http://protegewiki.stanford.edu/wiki/OntologyBeanGenerator`), que dá suporte à geração de arquivos Java representando uma ontologia que pode ser usada na plataforma JADE, usada neste trabalho e apresentada no Apêndice A.2. Através desse *plugin*, pode-se gerar inclusive os códigos de ontologia FIPA/JADE, facilitando bastante o desenvolvimento de ontologias para SMA. A Figura 2.8 mostra a interface da ferramenta *Protégé* com a ontologia do vinho.

Protocolo de Interação

O protocolo de interação é um dos pontos mais importantes no estudo de SMA. Sem interação, o trabalho dos agentes fica limitado e então, perde-se a razão para utilizar um SMA. Segundo Wooldridge (2002), a interação entre os agentes se dá basicamente de

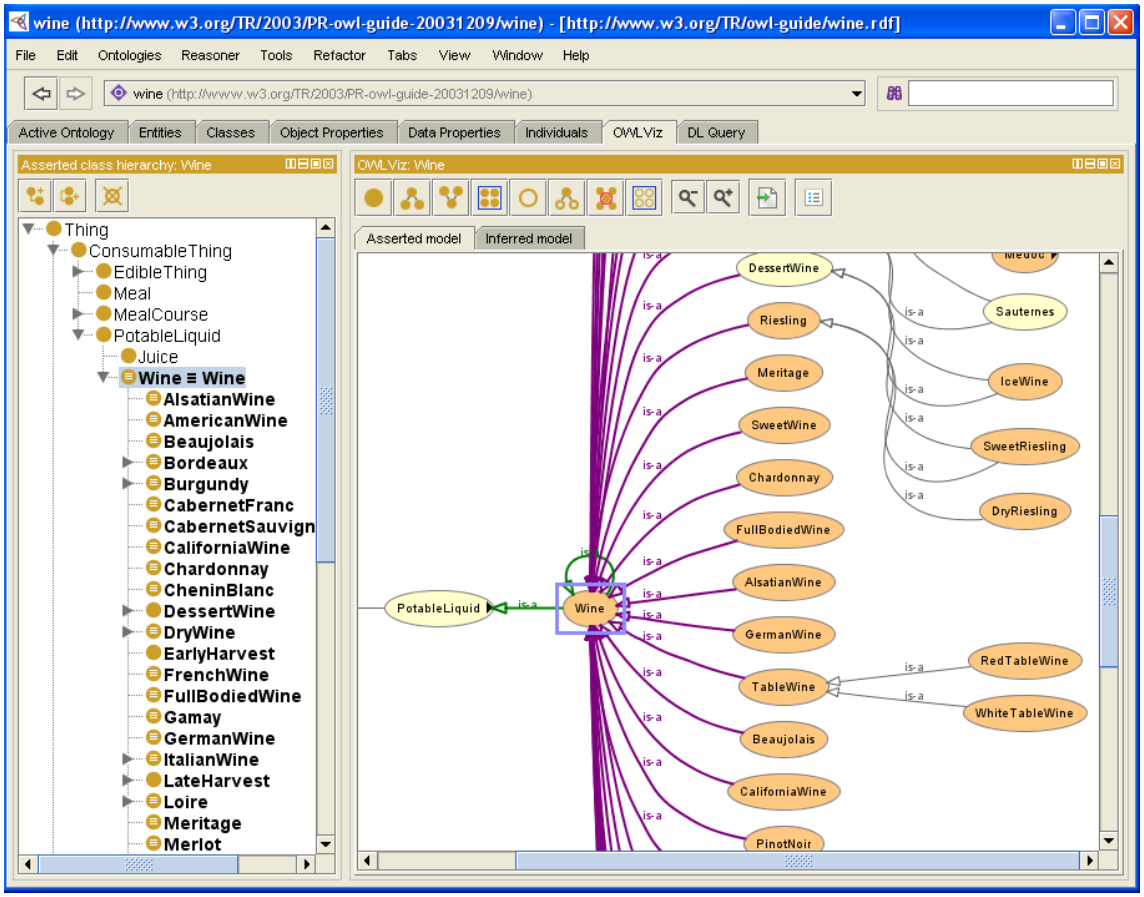


Figura 2.8: Inteface gráfica da ferramenta Protégé 4.0.2

duas formas: cooperativa ou competitiva. A forma cooperativa é quando todos os agentes trabalham juntos a fim de atingir um ou mais objetivos comuns, e assim, somam suas habilidades no processo. A forma competitiva é quando se tem diferentes agentes com diferentes objetivos. Eles podem então não compartilhar objetivos em comum, e assim, os agentes passam a usar estratégias para alcançar objetivos individuais, negociando recursos entre si. Ambientes assim são normalmente encontrado em jogos . Neste trabalho, utilizamos uma interação cooperativa entre os agentes.

Na literatura, dois aspectos essenciais para considerarmos em um SMA cooperativo são: a coerência e a coordenação. Segundo Wooldridge (2002), a coerência se refere ao quão bom o SMA se comporta como uma unidade. A coerência pode ser medida em termos da qualidade da solução, eficiência de utilização de recursos, entre outros. A coordenação é a habilidade do SMA dividir o objetivo em sub-objetivos e não permitir que os agentes acidentalmente se esbarrem nos sub-objetivos um dos outros, enquanto tentam atingir um objetivo em comum.

Um abordagem para solução do problema de coordenação dos agentes, segundo Bellifemine et al. (2007), é um criação de um plano do problema. Para evitar inconsistência e ações conflituosas, os agentes podem construir um plano detalhado de todas as futuras ações e interações requeridas para alcançar suas metas intercalando a execução com planos adicionais e replanejamentos. O plano pode ser centralizado ou distribuído. Num plano centralizado, há normalmente um agente coordenador que analisa os planos dos

agentes resolvendo conflitos e inconsistências. Já no planejamento distribuído, a idéia é prover a cada agente um modelo dos planos dos outros agentes. Nesse sentido, os agentes se comunicam para construir e atualizar seus planos individuais e dos outros agentes até resolver todos os conflitos.

Uma outra técnica de coordenação para alocação de tarefas e recursos entre agentes, baseada em estrutura organizacional é o protocolo *Contract Net*. Introduzido em Smith (1980) como um protocolo de alto nível para comunicação entre nodos em um resolvedor de problemas distribuído, o *Contract Net* facilita o controle distribuído da execução de tarefas de forma cooperativa. Essa importante técnica de cooperação é baseada numa estrutura de mercado descentralizado onde os agentes podem atuar em dois papéis, um gerente e um contratante. Segundo Bellifemine et al. (2007), a premissa básica nessa forma de coordenação é que se um agente não pode resolver um problema usando suas habilidades individuais, ele decompõe o problema em subproblemas. O problema de associar os sub-problemas é resolvido por um mecanismo de contratação que consiste nos seguintes passos:

1. Anúncio de contrato pelo agente gerente;
2. Entrega de propostas pelos agentes interessados no contrato (*contractors*) em resposta ao anúncio;
3. Avaliação das ofertas enviadas pelos interessados e escolha do interessado (*contractor*) com a proposta mais apropriada.

Este protocolo foi incorporado na especificação em FIPA (2002c). A Figura 2.9 mostra os passos para contratação de um serviço através do protocolo *Contract Net*. O agente Iniciador (*Iniciator*), solicita m propostas de outros agentes enviando uma mensagem do tipo *cfp* (*call for proposals*), que especifica a tarefa e as condições de execução. Os participantes que recebem a chamada, são vistos como potenciais contratados (*contractors*), e estão habilitados a gerar n respostas. Dessas respostas, j são propostas para executar a tarefa e i são respostas rejeitando participar do "leilão". Passado o tempo máximo (*deadline*), o Iniciador avalia as j propostas recebidas e seleciona um ou mais agentes para executar a tarefa (l). Depois que o selecionado completou a tarefa, ele envia uma mensagem do tipo *inform-done* ou *inform-result* comunicando ao Iniciador. Caso não tenha sido possível completar a tarefa, é enviada ao Iniciador uma mensagem de falha do tipo *failure* (Bellifemine et al., 2007).

2.2.4 Metodologias

Segundo Giorgini et al. (2002), há diversas razões que explicam por que o desenvolvimento de software orientado a agentes tem se tornado popular. Nos últimos anos houve um crescimento explosivo de comércio eletrônico, planejamento de recursos e computação móvel. Neste sentido, softwares devem ser agora baseados em arquiteturas mais flexíveis e que possam acomodar novos componentes e requisitos. Além disso, um software deve também operar em diferentes plataformas, sem necessitar ser recompilado, apresentando características, tais como: robustez e autonomia.

Atualmente, há diversas metodologias de desenvolvimento de software orientado a agentes. Sendo as mais citadas na literatura: Gaia, Tropos, Prometheus, MaSE e PASSI (Henderson-Sellersa e Giorgini, 2005).

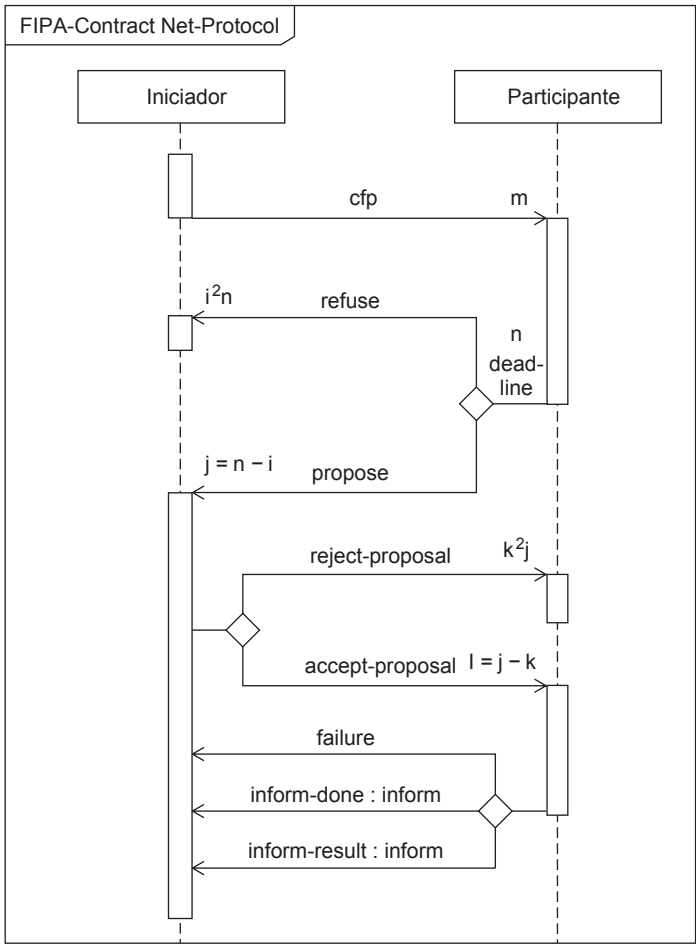


Figura 2.9: Diagrama do *FIPA Contract Net Protocol* (Bellifemine et al., 2007)

Neste trabalho utilizamos a metodologia Tropos, apresentada em Bresciani et al. (2004), por ser uma metodologia simples e focar principalmente na análise de requisitos iniciais e finais, sendo flexível quanto a utilização de *Unified Modeling Language* (UML). A Tropos é definida em termos dos conceitos de agente, objetivo e as noções mentalísticas relacionadas (conhecimento, crença, intenção, obrigação, entre outros). Essas noções são usadas para auxiliar todas as fases de desenvolvimento de software, desde a análise dos requisitos inicias até a implementação. Além disso, um papel crucial é dado à análise dos requisitos iniciais que precede à especificação de requisitos do sistema. Assim, Tropos auxilia as fases inicias do processo de desenvolvimento de software, mais do que as demais metodologias orientadas a objetos ou mesmo orientada a agentes, auxiliando na compreensão do formato do sistema com respeito à visão de agentes e suas metas, uma vez que na fase de requisitos iniciais são analisados processos que envolvem múltiplos participantes, sejam eles humanos ou agentes de softwares, e suas intenções (Giorgini et al., 2005).

Segundo Dam e Winikoff (2003), uma das diferenças significantes entre Tropos e as demais metodologias é seu forte foco na análise dos requisitos iniciais, onde os colaboradores do domínio e suas intenções são identificadas e analisadas. Este processo de análise permite que se capture a razão para desenvolver o software.

Segundo Giorgini et al. (2005), há quatro fases do desenvolvimento de software abrangida por Tropos: (i) análise dos requisitos iniciais, relativa à compreensão do problema através do estudo da configuração organizacional; (ii) análise dos requisitos finais, onde o sistema é descrito com seu ambiente operacional; (iii) projeto arquitetural, onde a arquitetura global do sistema é definida em termos de subsistemas, interconectados através de dados, controle e outras dependências; e por fim, (iv) projeto detalhado, onde os comportamentos de cada componente de software é definido de forma mais detalhada.

Na fase de Requisitos Iniciais, o Tropos usa o conceito de ator e metas para modelar os participantes do processo e suas intenções. Dentre as metas que podem ser definidas em Tropos, há um tipo específico chamado *softgoal*. Este tipo de meta é usado quando não há um critério claro para sua realização. Segundo Dam e Winikoff (2003), pode ser considerado o mapeamento de requisitos não funcionais. Dois diagramas são usados nesta fase, segundo Giorgini et al. (2005): diagrama de dependências e diagrama de ator.

O diagrama de dependências é um grafo envolvendo atores que tem dependências estratégicas entre si. Uma dependência representa um acordo entre dois atores e, nessa ligação, é utilizada uma seta que inicia no ator dependente e termina apontando no ator cujo primeiro depende para realização do objeto da dependência, seja ela uma meta, uma tarefa ou um recurso. Na Figura 2.10 pode-se observar um diagrama de dependências para uma loja de música. Os atores são respresentados por círculos, as metas por retângulos ovais, as metas fracas (*softgoals*) por nuvens, as tarefas por hexágonos e os recursos por retângulos.

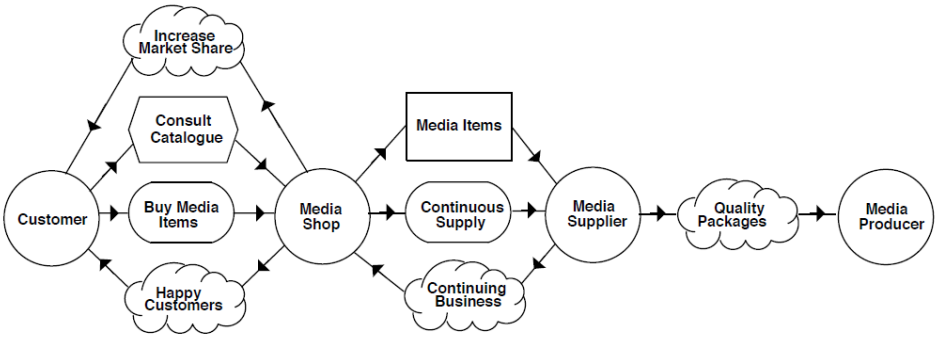


Figura 2.10: Diagrama de dependências para uma loja de música (Giorgini et al., 2005)

Um diagrama de ator aparece como um balão em que as metas para um específico ator são analisadas e as dependências com outros atores são estabelecidas. Metas são decompostas em sub-metas e as contribuições (positivas ou negativas) das sub-metas às suas metas. Na Figura 2.11, pode-se notar as metas e sub-metas do ator *Media Shop*. Os sinais +, ++, - e - - nas setas de dependência representam o nível de cooperação entre os objeto. Por exemplo, a meta "pedir pela internet" (*order by internet*) coopera com a meta "reduzir custos" (*reduce cost*), enquanto que a meta "aumentar preços das vendas" (*increase sales price*) concorre com a meta "deixar clientes felizes" (*happy customers*) (Giorgini et al., 2005).

Na fase de Requisitos Finais, segundo Dam e Winikoff (2003), os modelos que foram criados na fase de Requisitos Inicias são estendidos. A importância dessa fase é a modelagem do sistema alvo com seu ambiente. Os mesmos diagramas são utilizados nessa fase,

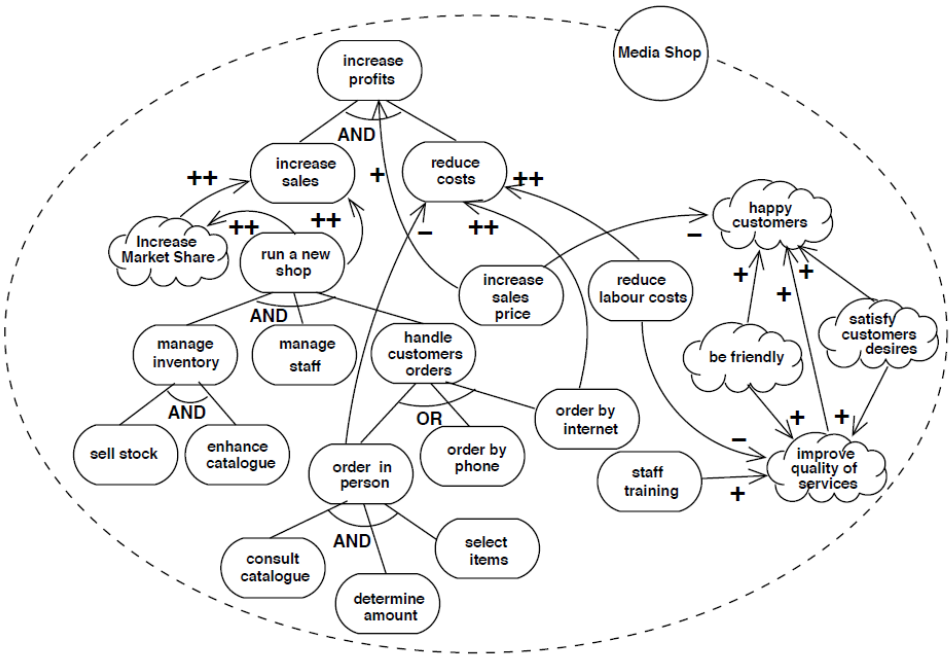


Figura 2.11: Diagrama do ator Media Shop (Giorgini et al., 2005)

no entanto, o foco é como o sistema deve ser. Sendo assim, novos diagramas são construídos com maior nível de detalhamento, inserindo metas mais específicas, e decompondo-as a fim de modelar melhor as funcionalidades do sistema.

Segundo Bresciani et al. (2004), na fase de Projeto Arquitetural, é definida a arquitetura global do sistema, em termos dos subsistemas (atores) interconectados através de dados em fluxos de controle (dependências). Esta fase é articulada em três passos. No primeiro, toda a organização arquitetural é definida. Novos atores são introduzidos no sistema como resultado de análises feitas em diferentes níveis de abstração. O segundo passo consiste na identificação dos recursos e capacidades necessários aos atores para realização de suas metas e planos (tarefas). No último passo, é definido um conjunto de tipos de agentes e associado a cada tipo uma ou mais capacidades.

Na fase de Projeto Detalhado, são feitas as especificações dos agentes num nível mais detalhado, tais como: comunicação entre agentes, capacidades, crenças e metas. Segundo Bresciani et al. (2004), abordagens práticas para essa atividade são normalmente propostas com plataformas de desenvolvimento específicas e normalmente depende das funcionalidades da linguagem de programação dos agentes adotada. Por exemplo, FIPA e *Object Management Group* (OMG) suportam a UML como linguagem que habilitada para especificação de sistemas de agentes. O bem conhecido protocolo de interação de agentes *Contract Net* é modelado em UML. Assim, durante esta fase, Tropos utiliza diagramas UML de atividades para representação das tarefas e capacidades dos agentes.

Para este trabalho, a escolha de Tropos justifica-se por esta metodologia possuir uma rica representação dos requisitos iniciais e finais do sistema, o que possibilita uma modelagem mais clara dos agentes e de suas conexões através das dependências representadas por metas, tarefas e recursos. Além disso, esta metodologia é flexível em relação à utilização de UML e se adapta à linguagem de desenvolvimento do software escolhida. Pelo

fato desta proposta fazer uso da plataforma JADE e também de uma extensão do *framework* Weka, ambos escritos na linguagem Java, que é orientada a objetos, UML será utilizada para diagramação das estruturas de pacotes e classes, bem como das representações dinâmicas feitas através de diagramas de atividades. Isto é necessário pelo fato de estarmos integrando um outro ramo de estudo, que é o de descoberta de conhecimento através de MD, com a abordagem de SMA. A ferramenta Weka, utilizada neste trabalho, a qual provê uma biblioteca de algoritmos de MD, também provê estas funcionalidades na linguagem Java, utilizando a abordagem orientada a objetos.

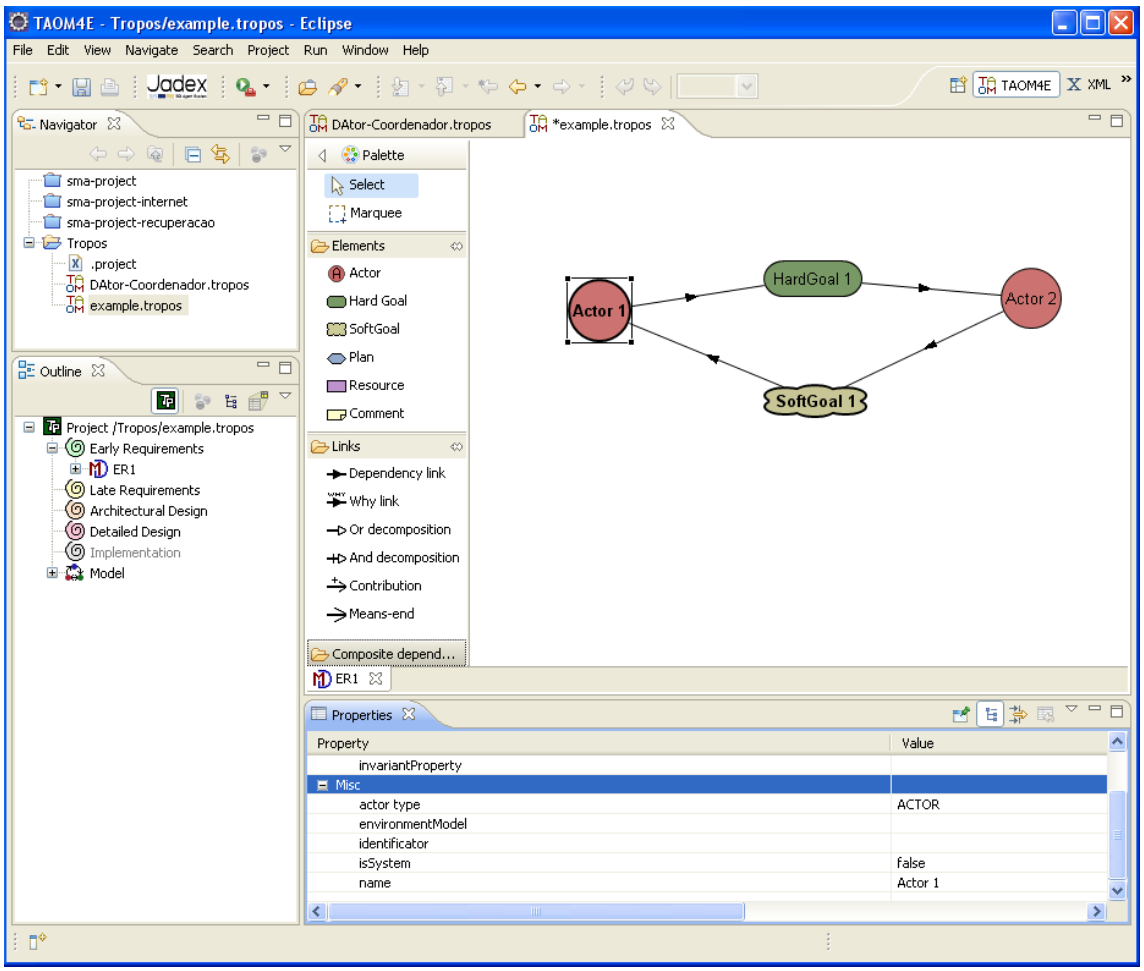


Figura 2.12: Captura de tela da plataforma Eclipse com o plug-in TAOM4E

Para confecção dos diagramas da metodologia Tropos, foi utilizada a ferramenta TAOM4E (`http://selab.fbk.eu/taom/`). A TAOM4E é software livre distribuída sob a licença GPL. Sendo um *plug-in* para a plataforma de desenvolvimento Eclipse, ferramenta adotada para desenvolvimento do protótipo. Essa integração é bastante interessante por possibilitar tanto a documentação, quanto o desenvolvimento do sistema, utilizando a mesma ferramenta. TAOM4E é bem documentada e sua utilização é demonstrada inclusive através de vídeos disponíveis no sítio da ferramenta. Além disso, sua instalação é fácil, sendo integrada automaticamente no Eclipse através da funcionalidade de adição automática de *plug-ins* já disponível na ferramenta. A Figura 2.12 apresenta a TAOM4E

na plataforma Eclipse. Em Bertolini et al. (2006) encontra-se maiores informações sobre a ferramenta TAOM4E.

Neste capítulo apresentamos os principais fundamentos teóricos utilizados neste trabalho de pesquisa, incluindo MD, DCBD e SMA. Passaremos a apresentar o contexto de aplicação de pesquisa na área de auditoria governamental, bem como apresentaremos alguns trabalhos correlatos.

Capítulo 3

Contexto de Aplicação

Este capítulo apresenta o contexto de aplicação do trabalho de pesquisa. Na Seção 3.1 serão apresentados os conceitos e princípios relacionados a Auditoria Governamental, necessários para o entendimento da aplicação da proposta; Enquanto a Seção 3.2 apresenta alguns trabalhos correlatos relacionado às áreas de pesquisa abordadas neste trabalho.

3.1 Auditoria Governamental

Segundo Jund (2005), há evidências de prática da atividade de auditoria 2600 anos A.C., na Suméria. No entanto, foi só a partir do século XV ou XVI, na Itália, que esta atividade começou a se consolidar no meio contábil. A auditoria moderna tem seu berço na Inglaterra, que a exportou para outros países, com o intuito de acompanhar seus investimentos. No caso do Brasil, na construção das estradas-de-ferro, já nos meados do século XIX. Apesar de já haver escritórios de auditoria no Brasil desde o início do século XX, a atividade de auditoria só foi reconhecida oficialmente no ano de 1986, por ato do Banco Central do Brasil.

Ferreira (2007) conceitua auditoria independente como a técnica contábil que tem por objeto, entre outros procedimentos, o exame, por um auditor independente, das atividades, livros e documentos de uma entidade, conforme a finalidade estabelecida num contrato de prestação de serviços de auditoria. Alguns exemplos de tarefas realizadas por auditorias independentes são:

- dar um parecer sobre as demonstrações contábeis exigidas por lei;
- realizar auditoria operacional para verificar desempenho da administração;
- identificar fraudes;
- verificar irregularidades.

No contexto governamental, a auditoria vem ganhando cada vez mais espaço devido aos tratados internacionais e até mesmo pelo próprio clamor social por maior transparência nos gastos públicos. O documento *Global Action Against Corruption*[1], lançado pela Organização das Nações Unidas (ONU) em 2004, reforça a necessidade da atuação dos

[1] O documento pode ser encontrado no sítio do UNODC (*United Nations Office on Drugs and Crime*) - `http://www.unodc.org/documents/corruption/publications_merida_e.pdf`

órgãos de auditoria governamental para promoção da transparência pública e o combate à corrupção. Desta forma, pode-se perceber uma tendência crescente de fortalecimento dos mecanismos de controle, que inclui a atividade de auditoria governamental, como esforços para não só combater a corrupção, mas também para melhorar a gestão pública como um todo, trazendo consequentes benefícios à população.

Auditoria Governamental, por sua vez, traz alguns outros elementos adicionais na tarefa de auditoria, devido ao próprio *modus operandi* da Administração Pública. Pelo princípio da legalidade, a Administração Pública só pode fazer o que a lei permite. No âmbito das relações entre particulares, incluindo as corporações em geral, o princípio aplicável é o da autonomia da vontade, que lhes permite fazer tudo o que a lei não proíbe (Pietro, 2009). Assim, a atividade de auditoria no contexto governamental deve levar em consideração as leis que regem o funcionamento da Administração Pública, para assim, agir de forma a atingir seus objetivos.

Na documento Proposta de Anteprojeto: Normas de Auditoria Governamental, elaborado pelo Tribunal de Contas do Estado da Bahia, a definição de auditoria governamental é (Bahia, 2007):

> Atividade independente e objetiva que, através da aplicação de procedimentos específicos, tem a finalidade de emitir opinião sobre a adequação das contas governamentais, assim como apresentar comentários sobre o desempenho organizacional e o resultado dos programas de governo. Além de exercer a fiscalização ou controle, a auditoria governamental também tem por objetivo agregar valor e otimizar o desempenho da Administração Pública, auxiliando o gestor público, no âmbito dos três Poderes e o Ministério Público, a alcançar seus objetivos, através de uma abordagem sistemática e disciplinada do controle da legalidade e legitimidade de seus atos e da avaliação da economicidade, da eficiência, eficácia, efetividade e eqüidade das ações governamentais sob sua responsabilidade.

Atualmente no Brasil, a atividade de auditoria governamental é realizada de duas formas: através do controles interno e externo, por determinação da Constituição Federal, que diz no seu Artigo 70 (Brasil, 1988):

> A fiscalização contábil, financeira, orçamentária, operacional e patrimonial da União e das entidades da administração direta e indireta, quanto à legalidade, legitimidade, economicidade, aplicação das subvenções e renúncia de receitas, será exercida pelo Congresso Nacional, mediante controle externo, e pelo sistema de controle interno de cada Poder

.

O sistema de controle interno, como o próprio nome já diz, não se trata de apenas um órgão, mas de um conjunto de elementos interconectados para realização da tarefa. No artigo 74 da Constituição, reza que todos os poderes manterão de forma integrada sistema de controle interno com a finalidade de:

1. avaliar o cumprimento das metas previstas no plano plurianual, a execução dos programas de governo e dos orçamentos da União;

2. comprovar a legalidade e avaliar os resultados, quanto à eficácia e eficiência, da gestão orçamentária, financeira e patrimonial nos órgãos e entidades da administração federal, bem como da aplicação de recursos públicos por entidades de direito privado;

3. exercer o controle das operações de crédito, avais e garantias, bem como dos direitos e haveres da União;

4. apoiar o controle externo no exercício de sua missão institucional.

3.1.1 A CGU

A CGU foi criada no dia 02/04/2001, pela Medida Provisória n° 2.143-31, chamada inicialmente de Corregedoria-Geral da União (CGU/PR) e sendo vinculada diretamente à Presidência da República. Teve como propósito original, o combate à corrupção e à fraude, e também a promoção da defesa do patrimônio público, no âmbito do Poder Executivo Federal.

Em 28/03/2002, através do Decreto n° 4.177, integrou a Secretaria Federal de Controle Interno (SFC) e a Comissão de Coordenação de Controle Interno (CCCI) à estrutura da então Corregedoria-Geral da União. No mesmo Decreto atribuiu-se também à Corregedoria-Geral da União as competências de Ouvidoria-Geral.

Atualmente, através do Decreto N° 4.304, de 16/07/2002, o papel de órgão central do Sistema de Controle Interno do Poder Executivo Federal é desempenhado pela CGU, que passou também a se chamar Controladoria-Geral da União, através da Lei n° 10.683 de 2003 [2].

Além de órgão central do Sistema de Controle Interno do Executivo, a CGU exerce também o papel de órgão central de Correição, sendo responsável por assistir direta e imediatamente ao Presidente da República quanto aos assuntos que, no âmbito do Poder Executivo, sejam relativos à defesa do patrimônio público e ao incremento da transparência da gestão, por meio das atividades de controle interno, auditoria pública, correição, prevenção e combate à corrupção e ouvidoria (Brasil, 2003).

Assim, a CGU se responsabiliza pelas atividades de auditoria governamental no âmbito do Poder Executivo Federal. Desenvolve, ainda, ações voltadas para a promoção da transparência e a prevenção da corrupção, que se destacam no núcleo essencial da proposta política e do programa de metas fundamentais do Governo Federal. Neste âmbito, a CGU tem se firmado também como uma típica agência anti-corrupção, que privilegia a elaboração de estratégias e políticas de prevenção e combate a esse mal (CGU, 2008).

Além disso, vinculada à Secretaria de Prevenção da Corrupção e Informações Estratégicas da CGU, está a Diretoria de Informações Estratégicas que atua dando suporte a atividades de pesquisa, produção e troca de informações de inteligência, com vistas a colaborar com as atividades das demais unidades da CGU, em especial na detecção de ilicitudes ocultas em atos, contratos e procedimentos administrativos. Essa diretoria colaborou na análise do conhecimento descoberto durante este trabalho de pesquisa através de especialistas da área. Os especialistas mencionados são profissionais que integram o quadro da CGU, e atuam na coleta e análise de informações estratégicas para o planejamento e execução de ações de controle tais como auditorias e fiscalizações públicas.

No contexto da CGU, a análise das licitações públicas nos processos de auditoria é de vital importância, pois a licitação é um meio comum de realização de despesa pública, além disso, o envolvimento de recursos financeiros possibilita e até mesmo incita a criação de esquemas ilícitos para manobrar a Lei, com finalidades diversas.

[2]Mais informações, vide `http://www.cgu.gov.br/CGU/Historico/index.asp`

3.1.2 Licitações Públicas

Segundo Pietro (2009), licitação é o procedimento administrativo pelo qual um ente público abre, a todos os interessados que se sujeitem às condições fixadas no instrumento convocatório, a possibilidade de formularem propostas dentre as quais selecionará e aceitará a mais conveniente para a celebração do contrato. Licitação pode então ser vista como um procedimento democrático para se contratar com o poder público. Este procedimento tem como objetivo garantir a observância do princípio constitucional da isonomia e selecionar a proposta mais vantajosa para a Administração (Brasil, 1993).

Dispõe a Constituição Federal, em seu artigo 37, XXI (Brasil, 1988):

> Ressalvados os casos especificados na legislação, as obras, serviços, compras e alienações serão contratados mediante processo de licitação pública que assegure igualdade de condições a todos os concorrentes, com cláusulas que estabeleçam obrigações de pagamento, mantidas as condições efetivas da proposta, nos termos da lei, o qual somente permitirá as exigências de qualificação técnica e econômica indispensáveis à garantia do cumprimento das obrigações.

Desta forma, para que a contratação pública seja viável, é dever da Administração Pública licitar.

Modalidades de Licitação

A Lei 8.666 é a lei que institui as normas para licitações e contratos da Administração Pública, a qual prevê cinco modalidades de licitação, no artigo 22: concorrência, tomada de preços, convite, concurso e leilão. A Lei nº 10.520 de 2002, cria também a modalidade de pregão. Passaremos a detalhar cada uma delas.

- Concorrência - segundo a Lei 8.666, destina-se à participação do maior número de concorrentes e é geralmente usada para contratações de maior vulto. A concorrência é obrigatória para compra, serviços gerais, obras e serviços de engenharia de valor superior ao fixado por lei federal, compra e alienação de bens imóveis, independente do valor, concessão de direito real de uso, licitações internacionais, entre outros.
- Tomada de Preços - envolve contratos de valor médio. Essa modalidade é realizada entre interessados previamente inscritos em cadastros administrativos ou que preencham os requisitos de cadastramento até o terceiro dia anterior à data do recebimento das propostas.
- Convite - é utilizado para contratações de pequeno valor. Nesta modalidade, a carta-convite substitui o edital. Deve ser realizada tendo no mínimo três interessados do ramo pertinente ao objeto, podendo participar aqueles que não receberam a carta-convite, porém, cadastrados com antecedência de 24 horas da apresentação das propostas.
- Concurso - é usada para escolha de trabalhos técnicos, científicos ou artísticos, mediante a instituição de prêmio ou remuneração aos vencedores.
- Leilão - utilizado conforme a Lei, para a venda de bens móveis inservíveis para a Administração ou de produtos legalmente apreendidos ou penhorados, ou para a

alienação de bens imóveis prevista no artigo 19 da Lei 8.666, a quem oferecer o maior lance, igual ou superior ao valor da avaliação.

- Pregão - na definição de Pietro (2009), é a modalidade de licitação para aquisição de bens e serviços comuns, qualquer que seja o valor estimado da contratação, em que a disputa pelo fornecimento é feita por meio de propostas e lances em sessão pública. A Lei 10.520/2002 permite que o pregão seja realizado por meio da utilização de recursos de tecnologia de informação, nos termos de regulamentação específica.

O Decreto nº 5.450/2005, que regulamenta a realização do Pregão Eletrônico, dispõe que essa modalidade de licitação realizar-se-á quando a disputa pelo fornecimento de bens ou serviços comuns for feita à distância em sessão pública, por meio de sistema que promova a comunicação pela Internet.

Além disso, o Decreto atribui a condução das licitações nessa modalidade ao órgão ou entidade promotora da licitação, com apoio técnico e operacional da Secretaria de Logística e Tecnologia da Informação do Ministério do Planejamento, Orçamento e Gestão, que atuará como provedor do sistema eletrônico para os órgãos integrantes do Sistema de Serviços Gerais (SISG). Como parte desse sistema, encontra-se o ComprasNet. O ComprasNet é um site WEB, instituído pelo Ministério do Planejamento, Orçamento e Gestão (MPOG), para disponibilizar, à sociedade, informações referentes às licitações e contratações promovidas pelo Governo Federal, bem como permitir a realização de processos eletrônicos de aquisição. É um módulo do Sistema Integrado de Administração de Serviços Gerais (SIASG), composto, atualmente, por diversos subsistemas com atribuições específicas voltadas à modernização dos processos administrativos dos órgãos públicos federais integrantes do SISG (`www.comprasnet.gov.br`).

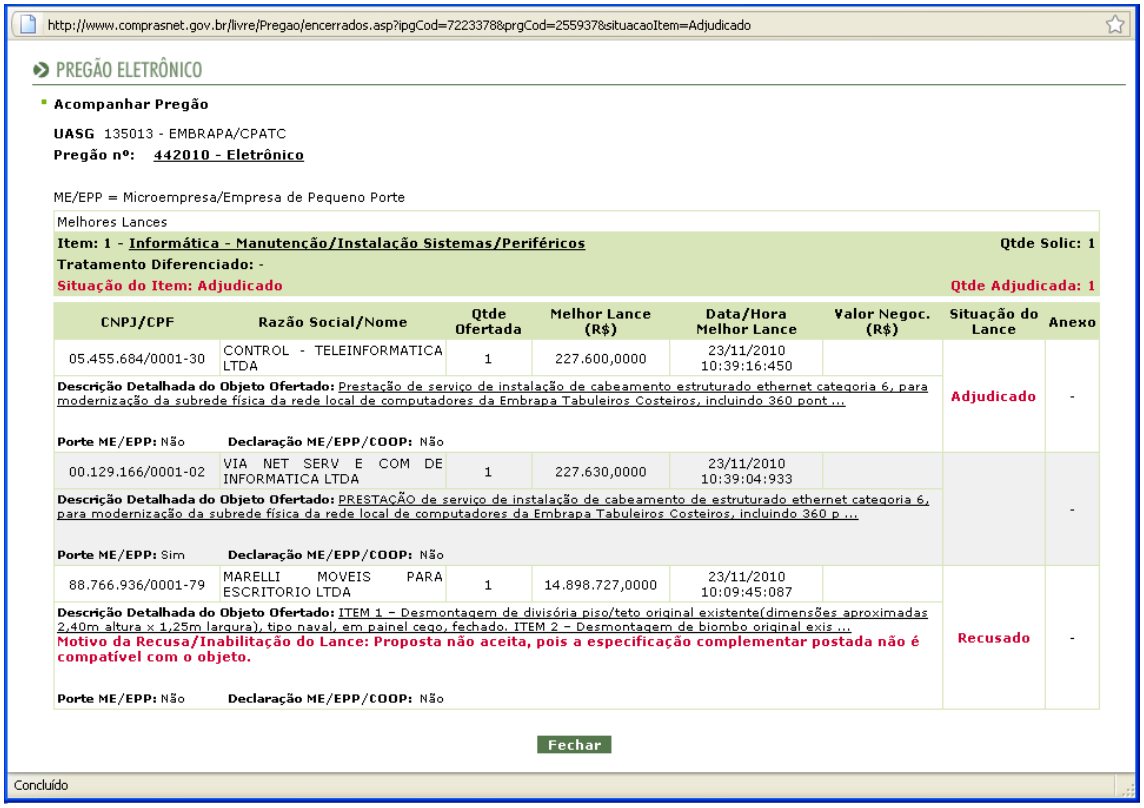
A Lei 8.666 define também os tipos de licitação, que definem como será escolhido o vencendor da licitação. Os tipos são: menor preço, melhor técnica, técnica e preço, e maior lance ou oferta (no caso de alienação de bens).

Na Figura 3.1, pode-se observar o resultado de um pregão realizado pela Embrapa (Empresa Brasileira de Pesquisa Agropecuária), para contratação de serviço de informática, do tipo "menor preço". Nota-se que o objeto em questão já foi adjudicado ao vencedor, no caso, a empresa *CONTROL - TELEINFORMATICA LTDA*, que apresentou o menor lance, de R$ 227.600,00. Nota-se também que a empresa *MARELLI MOVEIS PARA ESCRITORIO LTDA* apresentou lance com objeto ofertado incompatível ao objeto especificado na licitação. Desta forma, sua proposta não foi aceita, e o lance foi recusado. A diferença entre as duas melhores propostas, que definiu o vencedor foi de R$ 30,00, um valor ínfimo em relação ao vulto da compra.

Irregularidades em Licitações

Como a licitação é a porta de entrada mais comum para realização da despesa pública, as atividades de auditoria governamental dão especial atenção à análise dos processos de licitação e contrato. Esta preocupação se dá pelo fato de que o envolvimento de recursos financeiros possibilita e até mesmo incita a criação de esquemas ilícitos para manobrar a Lei, com finalidades diversas.

Há também irregularidades que a princípio não trazem danos materiais ao processo, no entanto, configuram-se falhas na execução do mesmo. Normalmente, essas irregularidades



http://www.comprasnet.gov.br/livre/Pregao/encerrados.asp?ipgCod=7223378&prgCod=255937&situacaoItem=Adjudicado

PREGÃO ELETRÔNICO

Acompanhar Pregão

UASG 135013 - EMBRAPA/CPATC
Pregão nº: 442010 - Eletrônico

ME/EPP = Microempresa/Empresa de Pequeno Porte

Melhores Lances

Item: 1 - Informática - Manutenção/Instalação Sistemas/Periféricos — **Qtde Solic: 1**
Tratamento Diferenciado: -
Situação do Item: Adjudicado — **Qtde Adjudicada: 1**

CNPJ/CPF	Razão Social/Nome	Qtde Ofertada	Melhor Lance (R$)	Data/Hora Melhor Lance	Valor Negoc. (R$)	Situação do Lance	Anexo
05.455.684/0001-30	CONTROL - TELEINFORMATICA LTDA	1	227.600,0000	23/11/2010 10:39:16:450		Adjudicado	-
Descrição Detalhada do Objeto Ofertado: Prestação de serviço de instalação de cabeamento estruturado ethernet categoria 6, para modernização da subrede física da rede local de computadores da Embrapa Tabuleiros Costeiros, incluindo 360 pont ...							
Porte ME/EPP: Não	**Declaração ME/EPP/COOP:** Não						
00.129.166/0001-02	VIA NET SERV E COM DE INFORMATICA LTDA	1	227.630,0000	23/11/2010 10:39:04:933			-
Descrição Detalhada do Objeto Ofertado: PRESTAÇÃO de serviço de instalação de cabeamento de estruturado ethernet categoria 6, para modernização da subrede física da rede local de computadores da Embrapa Tabuleiros Costeiros, incluindo 360 p ...							
Porte ME/EPP: Sim	**Declaração ME/EPP/COOP:** Não						
88.766.936/0001-79	MARELLI MOVEIS PARA ESCRITORIO LTDA	1	14.898.727,0000	23/11/2010 10:09:45:087		Recusado	-
Descrição Detalhada do Objeto Ofertado: ITEM 1 - Desmontagem de divisória piso/teto original existente(dimensões aproximadas 2,40m altura x 1,25m largura), tipo naval, em painel cego, fechado. ITEM 2 - Desmontagem de biombo original exis ... **Motivo da Recusa/Inabilitação do Lance: Proposta não aceita, pois a especificação complementar postada não é compatível com o objeto.**							
Porte ME/EPP: Não	**Declaração ME/EPP/COOP:** Não						

Fechar

Concluído

Figura 3.1: Detalhe dos lances de um pregão eletrônico no site ComprasNet

são classificadas como falhas formais. As irregularidades em processos de licitação se configuram pela não observação de algum item da Lei de Licitações, seja a quebra de um dos princípios, a não observação da formalidade do processo, entre outros. Uma das irregularidades comuns nos processos de licitação, é a chamada *Fracionamento de Despesas* em licitação. Essa irregularidade consiste basicamente em dividir a despesa para utilizar a modalidade de licitação inferior à determinada pela Lei, ou mesmo para realização da dispensa de licitação em razão do valor.

A Lei 8.666 no artigo 24 estipula hoje que para serviços e compras de valor até R$8.000,00 a licitação é dispensável. O fracionamento de despesa, pode ser ilustrado pela seguinte situação. Um gestor necessita adquirir 10 notebooks para sua unidade. O preço médio da compra é de R$15.000,00. Como a lei dispensa a licitação para valores até R$8.000,00, o gestor resolve comprar 5 computadores numa loja, e os outros 5 computadores em outra loja. Foram 2 compras diferentes, no entanto, o objeto em questão permanece o mesmo. No mesmo artigo 24 da Lei 8.666, há uma ressalva dizendo que a licitação é dispensada:

> [...] desde que não se refiram a parcelas de um mesmo serviço, compra ou alienação de maior vulto que possa ser realizada de uma só vez.

Esta é uma irregularidade que já foi inclusive objeto de vários deliberações do TCU [3]. Acórdão 1386/2005, Segunda Câmara diz:

[3] `http://portal2.tcu.gov.br/portal/page/portal/TCU/comunidades/licitacoes_contratos/10%20Fracionamento%20de%20Despesa.pdf`

> Evite a fragmentação de despesas, caracterizada por aquisições freqüentes dos mesmos produtos ou realização sistemática de serviços da mesma natureza em processos distintos, cujos valores globais excedam o limite previsto para dispensa de licitação a que se referem os inciso I e II do art. 24 da Lei 8.666/1993.

No mesmo sentido, o Acórdão 740/2004, Plenário do TCU, recomenda da seguinte forma:

> Planeje adequadamente as aquisições e/ou contratações a fim de evitar o fracionamento da despesa, em observância ao art. 23, §5º, da Lei nº 8.666/1993.

E vários outros Acórdãos, como 2528/2003 (Primeira Câmara), 1025/2003 (Plenário), 73/2003 (Segunda Câmara), 76/2002 (Segunda Câmara), tratam sobre o assunto no sentido de evitar ações que levem ao fracionamento de despesas. É uma irregularidade que tanto pode ser cometida com o objetivo de acelerar o processo de aquisição, quanto de evitar a concorrência com fins espúrios.

Outras irregularidades também ocorrem como:

- Desrespeito ao prazo mínimo para apresentar as propostas
- Nota de Empenho anterior à apresentação da proposta
- Proposta entregue antes da publicação do edital
- Possibilidade de competição em inexigibilidades
- Vículo entre licitante e servidores públicos da comissão de licitação

Ocorrências como as acima citadas configuram-se irregularidades por infringir os princípios da legalidade, publicidade, moralidade, entre outros, e possibilitam a criação de ambientes propícios à fraudes e corrupção. Outra irregularidade que pode ser encontrada em licitações é o chamado Rodízio em Licitações, que será descrito a seguir.

3.1.3 Rodízio em Licitações

Na Lei 8.666, em seu artigo 3º, são citados oito princípios básicos que devem reger as licitações públicas: legalidade, da impessoalidade, da moralidade, da igualdade, da publicidade, da probidade administrativa, da vinculação ao instrumento convocatório e do julgamento objetivo.

Entre os diversos princípios da licitação, destaca-se o princípio da igualdade, que segundo Pietro (2009):

> Constitui um dos alicerces da licitação, na medida em que esta visa, não apenas permitir à Administração a escolha da melhor proposta, como também assegurar igualdade de direito a todos os interessados em contratar. Esse princípio, que hoje está expresso no artigo 37, XXI, da Constituição, veda o estabelecimento de condições que impliquem preferência em favor de determinados licitantes em detrimento dos demais

.

Ainda segundo Pietro (2009), está implícito outro princípio da licitação no parágrafo 1º, inciso I, do artigo 3º da Lei nº 8.666, que é o da competitividade, decorrente do princípio da isonomia. A Lei diz que é vedado aos agentes públicos:

> Admitir, prever, incluir ou tolerar, nos atos de convocação, cláusulas ou condições que comprometam, restrinjam ou frustrem o seu caráter competitivo e estabeleçam preferências ou distinções em razão da naturalidade, da sede ou domicílio dos licitantes ou de qualquer outra circunstância impertinente ou irrelevante para o específico objeto do contrato.

O caráter competitivo da licitação é um dos fundamentos mais importantes desse ato, pois possibilita à Administração Pública, realizar suas compras a preços justos. A falta de competitividade num processo de licitação do tipo "menor preço", permite que uma empresa contrate com a Administração a preços superfaturados. Pois, se não há outra proposta, logo, a única proposta será a melhor. É fato que a Lei prevê casos excepcionais, onde não se é possível ter concorrência. Mas no caso comum, a concorrência e a competitividade trazem benefícios ímpares para o processo licitatório, e a falta desse princípio, permite a ocorrência dos chamados Cartéis, que trazem diversos danos ao Erário.

Segundo Ministério da Justiça (2009), cartel é um acordo explícito ou implícito entre concorrentes para principalmente fixação de preços ou quotas de produção, divisão de clientes e de mercados de atuação. O objetivo é, por meio da ação coordenada entre concorrentes, eliminar a concorrência, com o consequente aumento de preços e redução de bem-estar para o consumidor. O Conselho Administrativo de Defesa Econômica (CADE) é a Autarquia responsável por investigar e punir as empresas que se unem na prática do cartel. Esta prática configura tanto ilícito administrativo punível pelo CADE, nos termos da Lei nº 8.884/94, quanto crime, punível com pena de 2 a 5 anos de reclusão, nos termos da Lei nº 8.137/90.

O rodízio de licitações se dá quando um grupo de empresas forma um cartel com a finalidade de dividir as licitações entre si elevando o preço de contratação com a Administração Pública, trazendo consequentemente danos ao Erário.

Segundo Araujo e Martinez (2010), os dados da Organização para a Cooperação e Desenvolvimento Econômico (OCDE) mostraram que os cartéis em licitações geram um sobrepreço mínimo 20% comparado ao preço em um mercado competitivo, causando perdas anuais de centenas de bilhões de reais ao Erário e contribuintes, que deixam de se beneficiar de outros serviços e produtos que poderiam ser contratados com o pagamento de seus impostos que foram ilicitamente desviados. Sem o apelo da pena privativa de liberdade, o Estado não consegue atingir o efeito dissuasório necessário para evitar que novos cartéis se formem em detrimento da sociedade.

Ainda citando Araujo e Martinez (2010), países desenvolvidos gastam em média 15% do seu PIB em compras públicas, sendo que esse percentual é maior em países em desenvolvimento, ainda segundo dados da OCDE. Os cartéis em licitação usualmente ocorrem em conjunto a outros crimes, como corrupção, lavagem de dinheiro e evasão fiscal, o que reforça a necessidade de uma robusta rede de inteligência. Em 2009, a Secretaria de Direito Econômico firmou acordos de cooperação com a CGU e o Tribunal de Contas da União (TCU) que, somados aos acordos existentes com a Polícia Federal e locais, Ministério Público Federal e Estaduais, possibilitará uma atuação coordenada no território nacional para a adequada repressão ao ilícito.

Licitações são um ambiente propício à atuação dos cartéis, que podem agir de várias formas. A seguir, são listados alguns comportamentos que podem ser utilizados para identificação de cartéis em licitações (Justiça, 2008).

- Fixação de preços - há um acordo firmado entre concorrentes para aumentar ou fixar preços e impedir que as propostas fiquem abaixo de um "preço base".
- Direcionamento privado da licitação - há a definição de quem irá vencer determinado certame ou uma série de processos licitatórios, bem como as condições nas quais essas licitações serão adjudicadas.
- Divisão de mercado - representada pela divisão de um conjunto de licitações entre membros do cartel, que, assim, deixam de concorrer entre si em cada uma delas. Por exemplo, as empresas A, B e C fazem um acordo pelo qual a empresa A apenas participa de licitações na região Nordeste, a empresa B na região Sul e a empresa C na região Sudeste.
- Supressão de propostas - modalidade na qual concorrentes que eram esperados na licitação não comparecem ou, comparecendo, retiram a proposta formulada, com intuito de favorecer um determinado licitante, previamente escolhido.
- Apresentação de propostas "pro forma" - caracterizada quando alguns concorrentes formulam propostas com preços muito altos para serem aceitos ou entregam propostas com vícios reconhecidamente desclassificatórios. O objetivo dessa conduta é, em regra, direcionar a licitação para um concorrente em especial.
- Rodízio - acordo pelo qual os concorrentes alternam-se entre os vencedores de uma licitação específica. Por exemplo, as empresas A, B e C combinam que a primeira licitação será vencida pela empresa A, a segunda pela empresa B, a terceira pela empresa C e assim sucessivamente.
- Sub-contratação - concorrentes não participam das licitações ou desistem das suas propostas, a fim de serem sub-contratados pelos vencedores. O vencedor da licitação a um preço supra-competitivo divide o sobre-preço com o subcontratado.

Em muitos cartéis, mais de uma dessas formas de atuar podem estar presentes. Assim, a prática do "rodízio" pode ser combinado com a divisão de mercado (os concorrentes combinam a alternação dos vencedores em um grupo de licitações, para dar a impressão de efetiva concorrência), e o direcionamento da licitação pode ser implementado pela apresentação de propostas inviáveis e complementado por subcontratação. De qualquer forma, o resultado sempre é o aumento dos preços pagos pela Administração e a conseqüente transferência ilegítima de recursos para os membros do cartel (Justiça, 2008).

Em Justiça (2008) podemos ainda encontrar diversos indícios de possível rodízio em licitações. Algumas características são:

- As propostas apresentadas possuem redação semelhante ou os mesmos erros e rasuras.
- Certos fornecedores desistem, inesperadamente, de participar da licitação.
- O valor das propostas se reduz significativamente quando um novo concorrente entra no processo (provavelmente não integrante do cartel).
- Um determinado concorrente vence muitas licitações que possuem a mesma característica ou se referem a um tipo especial de contratação.

- Existe um concorrente que sempre oferece propostas, apesar de nunca vencer as licitações.

Essas características, para serem observadas, necessitam muitas vezes de análises complexas em base de dados, pois a detecção em tempo real de um possível rodízio em licitação atuando num determinado processo é inviável, devido às manobras normalmente realizada pelos participantes do cartel, que sabem que estão cometendo um ato ilícito. Desta forma, no trabalho de detecção e investigação de cartéis, a SDE depende muito das informações e denúncias advindas de outras autoridades ou partes prejudicadas. São essas fontes de informação, muitas vezes anônimas, que ajudam a Secretaria a tomar ciência das infrações e que permitem a mobilização dos mecanismos disponíveis para descobrir e fazer cessar condutas anticoncorrenciais (Justiça, 2008).

A CGU como Órgão Central de Controle Interno, mantém parceria com o CADE para que as investigações de prática de cartéis no âmbito da Administração Pública sejam mais eficientes. Desta forma, sempre que a CGU encontra indícios de práticas de Rodizío de Licitações em suas auditorias, o processo pode ser encaminhado ao CADE para que este tome as providências cabíveis. Independente da decisão do CADE, pode também a CGU punir as empresas suspeitas através da Declaração de Inidoneidade, que as impede imediatamente de licitar e contratar com a Administração Pública.

Apresentaremos alguns trabalhos relacionados a proposta deste trabalho com foco na integração de MD e SMA.

3.2 Trabalhos Correlatos

Considerando a missão da CGU e suas responsabilidades, há uma série de problemas relacionados a corrupção, tais como fraudes corporativas, peculato, fracionamento de licitações, direcionamento de editais de licitações para determinadas empresas, cartéis, entre outros. Isso tem emergido como um desafio para centros econômicos em grande crescimento como o Brasil, Rússia, Índia e China. Um documento importante feito por mais de 75 peritos, que examinaram a escala, escopo e consequências devastantes de corrupção é o Relatório de Corrupção Global da Transparência Internacional (*Transparency International's Global Corruption Report 2009*). Esta publicação documenta em detalhes os muitos riscos gerados pela corrupção para as empresas, desde os pequenos empreendimentos na África Sub-Saariana às multinacionais da Europa e América do Norte (contributors, 2009)[4].

Como citado, em contributors (2009), formação de cartel é um grave problema de corrupção que desafia pesquisadores em relação ao seu combate. Entretanto, até onde pesquisamos, não pudemos encontrar nenhum trabalho aplicado especificamente à detecção de cartéis em licitações públicas fazendo uso da integração de SMA e MD. Um *framework* de MD orientado a agentes é introduzido em Zghal et al. (2005), porém projetado para trabalhar apenas com dados espaciais. Podemos citar outros trabalhos que propõe uma abordagem de integração entre SMA e MD aplicada a domínios tais como Bioinformática, Análise de Dados em Rede, Processamento de Vídeos (Karoui et al., 2009; Lee, 2009; Ralha et al., 2008).

[4]Para maiores detalhes, vide `http://www.cgu.gov.br/conferenciabrocde/arquivos/English-Global-Corruption-Report-2009.pdf`

Carvalho et al. (2009) apresenta o uso de *Probabilistic OWL* - (PR-OWL) para projetar e testar um modelo que executa fusão de informações para detectar possíveis fraudes em licitações envolvendo recurso Federal. Para projetar este modelo, uma ferramenta para criação de ontologia PR-OWL foi utilizada com ajuda de especialistas em PR-OWL e um especialista da CGU em detecção de fraudes. Os dados usados no estudo de caso são também oriundos da CGU, e representam vários editais de licitações expedidos pelos órgãos federais, estaduais e municipais. A idéia do trabalho é representar o conhecimento do especialista e então raciocinar através de ontologias probabilísticas a fim de transformar os itens de informações em itens de conhecimento através de inferência lógica e probabilística. O resultado é usado para traçar algumas conclusões sobre possíveis irregularidades, por exemplo, o direcionamento de editais de licitações para certas empresas. No entanto, no contexto específico de cartéis em licitações, não foi citada a possibilidade de aplicação do trabalho proposto na solução deste problema. Este tipo de irregularidade em licitações mostra-se mais apropriado para a utilização de técnicas de MD, uma vez que estas podem atuar identificando padrões para posteriormente serem avaliados pelo o especialista (Chapman et al., 2000). Nesse processo, o especialista irá atuar muito mais no sentido de reconhecer as características de cartéis nos modelos apresentados, do que fornecer conhecimento para geração dos modelos, o que seria um tanto oneroso, levando em conta que cada modelo é definido por um conjunto específico de empresas, gerando um espaço exponencial de soluções. Além disso, segundo o especialista a formação de cartéis normalmente não é documentada, dificultando assim a identificação de itens de informações que levem aos modelos através de inferência lógica e probabilística. Embora o trabalho não cite o uso de agentes, a abordagem seria útil em ambos os casos no sentido de possibilitar tanto a execução do processo quanto a manutenção dos modelos encontrados.

Gao et al. (2005) apresenta um modelo cooperativo para agentes de MD. Este trabalho tem abordagem de cooperação de diferentes agentes de MD e a combinação de conhecimento minerado em estruturas de conhecimento. O modelo usou dados médicos relacionados a pacientes com diabetes. Foram utilizados dois agentes de MD para produzir regras sequenciais e regras conjuntivas. Um outro agente foi usado para combinar as regras produzindo regras híbridas. Os resultados obtidos confirmaram a relação entre hipertensão e diabetes. No entanto a desempenho da aplicação do modelo não apresentou bons resultados, talvez pela utilização de algoritmos genéticos nos agentes. Além disso, a arquitetura apresentada não pareceu viável para ser aplicada no contexto de auditoria governamental, especialmente aplicada ao problema de detecção de cartéis, pelo fato de faltar na estrutura suporte a outros tipos de técnicas de MD, além de não apresentar um modelo adequado de coordenação e interação de agentes.

Karoui et al. (2009) propõe um *framework* multiagentes para detecção de anomalias em *firewalls* distribuídos integrando técnicas de MD na linha de pesquisa de AMII. Um conjunto de algoritmos e técnicas de MD são apresentados para analisar, gerenciar e detectar anomalias em regras de *firewalls* distribuídos com ajuda de agentes de softwares. Para isso, o autor utiliza um agente estático para executar as técnicas de MD no intuito de gerar novas políticas eficientes para os *firewall* da rede. Posteriormente, um agente móvel trafega por esses *firewalls* explorando essas regras com a finalidade de detectar eventuais anomalias no ambiente. A geração de regras é feita com ajuda do administrador de redes, que especifica os atributos que deverão ser utilizados para geração das políticas de segurança nos *firewalls* atraves dos agentes de MD. Técnicas de regras de associação,

agregação, e regressão são executadas de forma integrada pelo agente de MD para geração das políticas. A idéia do agente móvel atuando nos diversos *firewalls* da rede é interessante e poderá ser incorporada futuramente em nossa proposta. Nesse sentido, agentes de softwares seriam adpatados para gerar políticas a partir do conhecimento encontrado nas bases de auditoria para aplicação nos sistemas governamentais com finalidade preventiva. O protótipo apresentado pelo autor é voltado especificamente para o tratamento de políticas de segurança de *firewalls*, não permitindo a aplicação do modelo em outros contextos tais como auditoria.

Albashiri et al. (2009) apresenta EMADS, um sistema que utiliza uma arquitetura genérica de MD orientada por agentes. A arquitetura tem como base a plataforma JADE. Segundo o autor, a visão do sistema é de uma coleção anárquica de agentes criada através de contribuições da comunidade de usuários do EMADS espalhada pela internet, onde os agentes podem negociar uns com os outros na tentativa de realizar uma variedade de tarefas de MD. Uma demonstração do EMADS está atualmente em operação na Universidade de Liverpool. O *framework* conceitual proposto no trabalho é uma extensão do JADE e possibilita a criação de diversos tipos de agentes de softwares, inclusive de MD, tornando a abordagem bastante flexível. No entanto, o estudo apresentado mostrou a utilização do sistema em tarefas de regras de associação e classificação, sem contudo apresentar uma integração entre essas tarefas de MD. Além disso, a proposta de um ambiente aberto como a internet para processamento de dados de auditoria não é o mais apropriado, considerando a sensisibilidade dos dados. No contexto de auditoria, é necessária também uma proposta arquitetural que permita além da descoberta de conhecimentos, a adaptação de agentes para atuar não só em base de dados de auditoria, mas também nos sistemas governamentais, extrapolando assim a visão do EMADS.

No Capítulo 4 apresentaremos uma proposta de solução para o problema de cartéis em licitações utilizando técnicas de MD, e posteriormente integrando agentes de MD numa arquitetura distribuída no intuito de automatizar o processo de descoberta de conhecimento e melhorar a qualidade do conhecimento descoberto através da autonomia dos agentes.

Capítulo 4

Apresentação do Modelo de Solução

Uma resposta ao problema de correlacionar dados para se extrair conhecimento útil no âmbito de auditoria governamental pode ser dada através da utilização de técnicas de MD. No entanto, a aplicação de MD exige a compreensão dos dados e do negócio para que sejam aplicadas as técnicas mais apropriadas para extração dos padrões desejados, conforme citado nas primeiras fases do CRISP-DM, apresentado na Seção 2.1.1.

O caso específico da identificação de cartéis em licitações é uma tarefa complexa. Para confirmar grupos suspeitos é necessária a análise de vários processos de licitação extrapolando o escopo de apenas um órgão da Administração Pública. Além disso, cartéis podem atuar em várias cidades, e até mesmo em diferentes estados da Federação. Desta forma, a análise deste tipo de problema, na ótica da auditoria, volta-se para a averiguação dos dados correlacionados, pois assim, pode-se verificar de forma global a atuação dos grupos suspeitos. Mas como já foi mencionado, esta análise é feita quando já se tem os grupos suspeitos.

Análise de cruzamentos de dados para levantamento de evidências na procura de cartéis utilizando linguagens de consultas tais como SQL é impraticável, pois os espaço de soluções é exponencial, levando-se em conta que todas as combinações de empresas poderiam ser uma possível formação de cartel.

Quando se trata de uma base de dados de licitações, as informações básicas que compõem os registros são os participantes, que no caso são os chamados *fornecedores*, o objeto ou serviço que se está licitando, o órgão responsável, o local (município/UF) e o vencedor.

No caso do Sistema ComprasNet, onde são feitas as licitações na modalidade de Pregão do Governo Federal, são registrados os lances dados por cada fornecedor participante (conforme citado na Seção 3.1.1). A planilha manuseada pela CGU com essas informações consiste no registro dos últimos lances dos fornecedores participantes de cada licitação, sendo que o lance homologado diferente de zero é o lance que identifica o fornecedor vencedor da licitação. Um exemplo pode ser visto na Figura 3.1 e nos respectivos dados da Tabela 4.1. Neste caso, como as informações são consolidadas em uma única tabela, a normalização é perdida.

A procura de cartéis em licitações é a princípio a procura de empresas "associadas" com atuação frequente em processos de licitações. Outros aspectos são analisados, mas a característica de associação frequente é a mais forte na detecção de cartéis, uma vez que esta é a principal característica que vai de encontro ao princípio da livre concorrência em que se baseia a idéia de licitação (conforme apresentado na Seção 3.1.2). Assim, a técnica

Tabela 4.1: Amostragem de base fictícia de licitação

Licitação	Cód. Órgão	UF	Objeto	Fornecedor	Valor	Homologado
1	121	DF	Objeto X	fornecedor-A	10.00	10.00
1	121	DF	Objeto X	fornecedor-B	12.00	0.00
1	121	DF	Objeto X	fornecedor-D	12.50	0.00
2	133	MG	Objeto Y	fornecedor-A	42.00	0.00
2	133	MG	Objeto Y	fornecedor-E	41.50	0.00
2	133	MG	Objeto Y	fornecedor-F	40.00	40.00
2	133	MG	Objeto Y	fornecedor-G	43.00	0.00
3	121	DF	Objeto Z	fornecedor-C	21.00	0.00
3	121	DF	Objeto Z	fornecedor-B	18.75	0.00
3	121	DF	Objeto Z	fornecedor-A	18.10	18.10

de MD de Regras de Associação apresenta-se como promissora e adequada na solução deste problema (conforme apresentado na Seção 2.1.4).

4.1 Solução utilizando Regras de Associação

As Regras de Associação se propõem a descobrir relações fortes entre determinados atributos ou padrões em forma de regras que associam valores de atributos a um determinado conjunto de dados, como apresentado na Seção 2.1.4. No entanto, a aplicação de Regras de Associação na Tabela 4.1, por exemplo, irá procurar associação entre os atributos *Cód. Órgão*, *UF*, *Objeto*, *Fornecedor*, *Valor* e *Homologado*, associações estas irrelevantes na procura de cartel. É necessário adaptar a base para que seja possível associar os fornecedores entre si.

Poderia-se organizar a base de forma que os fornecedores participantes da licitação ficassem em colunas diferentes, sendo assim, diferentes atributos - *Participante 1*, *Participante 2*, *Participante 3*. Mas esta solução é inviável pois não se pode prever o número máximo de participantes de uma licitação. E mesmo que se pudesse prever, um determinado fornecedor participante de várias licitações, poderia aparecer em diferentes colunas na base para cada licitação de que ele participou. Não é possível garantir que o Fornecedor-A sempre apareça no campo *Participante 1*. Desta forma, a frequência desse atributo seria afetada quando o Fornecedor-A aparecesse em um campo diferente de *Participante 1*.

Definimos, então, uma estratégia de organização de dados para utilização da técnica de Regras de Associação, de forma que cada fornecedor da base de dados seja um atributo booleano no *dataset* preparado e cada instância do *dataset*, seja um processo de licitação. Assim, para cada licitação, o atributo relativo a um determinado fornecedor é preenchido com o valor *true*, caso aquele fornecedor tenha participado da licitação ou *false*, caso contrário.

Assim, a preparação dos *datasets* para regras de associação se resume em construir a matriz A formada por m linhas e $n + 1$ colunas, tal que:

$m =$ (número total de licitações da base de dados)
$n =$ (número total de fornecedores da base de dados)

$$a_{i,j} = \begin{cases} true & \text{se fornecedor } j \text{ participou da licitação } i; \\ false & \text{se fornecedor } j \text{ não participou da licitação } i; \end{cases}$$

$$1 \leq i \leq m; 1 \leq j \leq n;$$

$$a_{i,n+1} = vencedor(i)$$

$$1 \leq i \leq m; vencedor(i) = \text{CNPJ da empresa vencedora da licitação } i$$

Desta forma, as regras esperadas são do tipo:

$$fornecedor_A = true, \quad fornecedor_B = true \rightarrow vencedor = forncedor_C$$

O preenchimento da coluna de vencedores pode ser também eliminado, produzindo regras do tipo:

$$fornecedor_A = true, \quad fornecedor_B = true \rightarrow forncedor_C = true$$

Para obter regras associando apenas fornecedores, que é essencialmente o que queremos, o *dataset* gerado a partir dos dados da Tabela 4.1 pode ser visto na Tabela 4.2. As colunas {A,B,C,D,E,F,G}, representam a participação dos fornecedores {A,B,...,G} em cada processo de licitação da base de dados. Desta forma, o *dataset* terá como atributos todos os fornecedores da base de dados, possibilitando a busca de associações entre fornecedores com frequencia relevante na base de dados, através da técnica de Regras de Associação.

Tabela 4.2: Amostragem de nova base fictícia de licitação

Licitação	A	B	C	D	E	F	G
1	true	true	false	true	false	false	false
2	true	false	false	false	true	true	true
3	true	true	true	false	false	false	false

4.1.1 Teste da solução de MD utilizando Weka

Para testar a estratégia apresentada na Seção 4.1, foram realizadas atividades de mineração de dados numa base de licitações extraída do sistema ComprasNet. Os experimentos foram regidos segundo as disciplinas do modelo de referência CRISP-DM (Seção 2.1.1).

Os dados utilizados são relativos a todas as licitações para contratação de um mesmo tipo de serviço na modalidade de Pregão para órgãos do Poder Executivo Federal durante os anos de 2005 a 2008, em todos os estados da Federação. Os testes foram executados utilizando a ferramenta Weka em sua versão 3.6.1.

A Tabela 4.3 mostra alguns dados da base utilizada nos experimentos. Cada registro da base de dados representa o último lance de cada fornecedor participante de uma determinada licitação. Foram preparados dois *datasets*, seguindo a estratégia apresentada

Tabela 4.3: Base de dados utilizada nos experimentos preliminares

Informações	**Total**
Registros	26612
Licitações	2701
Empresas	3048
Empresas que já ganharam pelo menos 1 licitação	1162
Empresas que já ganharam pelo menos 5 licitações	121

na Seção 4.1, para aplicação da técnica de Regras de Associação. O algoritmo utilizado neste experimento foi o Apriori.

O primeiro *dataset* foi construído contemplando todas as licitações da base e todos os fornecedores. Já o segundo *dataset* contemplou apenas os fornecedores que já tinham participado de pelo menos duas licitações (Tabela 4.4), para buscar associação entre fornecedores que tinham um histórico de participação em licitações.

Como se pode observar na Tabela 4.2, quando se coloca todos os fornecedores como atributos, é mais comum encontrar valores *false* que valores *true*, pois não é razoável que todos os fornecedores de uma base de dados participem da maioria das licitações, visto que a base de dados tem como escopo licitações de todo o Brasil. Desta forma, é mais comum a não participação dos fornecedores. Por esta razão, é necessário suprimir as regras de associação que atestem associação de fornecedores com valores de não participação, como a regra a seguir:

$$Fornecedor - E = false,\ \ Fornecedor - F = false \rightarrow fornecedor - G = false$$

Desta forma, para suprimirmos este tipo de regra, a preparação dos *datasets* para Regras de Associação foi adaptada da seguinte forma: o preenchimento da matriz teve os valores de "não participação" de um determinado fornecedor num processo de licitação alterado de "false" para "?". No Weka, "?" é usado no lugar de valor faltante, sendo ignorado na execução dos algoritmos. Assim, nossos *datasets* foram preparados conforme a matriz a seguir.

$$a_{i,j} = \begin{cases} true & \text{se fornecedor } j \text{ participou da licitação } i; \\ ? & \text{se fornecedor } j \text{ não participou da licitação } i; \end{cases}$$
$$i \leq m; j \leq n;$$

A Tabela 4.4 mostra o resultado da execução do algoritmo Apriori nos *datasets* preparados. A escolha de valores altos na configuração do suporte mínimo para execução do algoritmo não nos garante boas regras para identificação de cartéis. Uma regra que associa alguns fornecedores e que tem suporte alto provavelmente indica a presença de grandes fornecedores participando de várias licitações. Desta forma, a configuração de um suporte mínimo alto para execução do algoritmo pode suprimir a aparição de diversas regras boas, com reais características de cartéis.

Valores altos de confiança, por sua vez, garantem a seleção de regras boas. Como as regras de associação são apresentadas no formato de uma implicação lógica, a regra terá um conjunto de fornecedores A no lado antecedente e um conjunto de fornecedores B no

Tabela 4.4: Resultados da execução do Apriori para os dois *datasets*

	Instâncias	**Atributos**	**Sup. Mín.**	**Conf. Mín.**	**Nº de Regras**
Dataset 1	2701	3049	1%	70%	294
Dataset 2	2370	1086	1%	80%	145

lado consequente, do tipo $A \rightarrow B$. No caso específico do problema de Rodízio de Licitações, o valor alto de confiança garante que a frequência de ocorrência dos fornecedores que figuram o lado antecessor da regra seja aproximada à frequência de ocorrência dos fornecedores que figuram no lado consequente da regra. Desta forma, todos os fornecedores que figuram na regra como um todo, podem ser considerados um grupo para fins de identificação de cartéis.

A decisão de reduzir o valor do suporte mínimo teve como consequência o aumento no número de regras encontradas, como mostra a Tabela 4.4, sendo necessária a definição de uma função de avaliação de regras para poder classificar e selecionar as melhores regras obtidas. Alguns problemas foram encontrados na aplicação dessa solução. As regras produzidas com suporte mínimo de 1% traziam regras com ocorrência mínima de 27 vezes na nossa base de dados. Desta forma, ficavam de fora regras de menor ocorrência, porém possivelmente de boa qualidade.

Na análise dos especialistas, era bem razoável que um cartel tivesse atuado de 10 a 15 vezes num espaço de tempo de três anos, o que não poderia ser captado no nosso procedimento de mineração de dados, por causa da configuração do suporte mínimo. Por outro lado, foi observado que também não era possível reduzir o quanto queríamos o valor do suporte mínimo, porque quanto menor esse valor era configurado, mais memória era consumida pelo algoritmo por causa do aumento do número de regras, ocasionando assim interrupção da execução por erro de estouro de memória. Sendo assim, foi necessária uma abordagem para contornar esses problemas, dividindo o espaço de soluções.

4.1.2 Técnica de Clusterização para divisão do espaço de soluções

Como a base de dados contém licitações de todo Brasil, uma abordagem era dividir a base por regiões macroeconômicas. O estudo do negócio possibilitou verificar que muitas vezes os fornecedores não se restringem necessariamente à regiões macroeconômicas. Um exemplo típico é a situação de Mato Grosso do Sul, Goiás e Tocantins. Embora Mato Grosso do Sul e Goiás pertençam à mesma região, é mais provável que os fornecedores do estado de Goiás atendam ao estado de Tocantins, por causa da proximidade geográfica, do que ao estado de Mato Grosso do Sul. O estado de Tocantins porém, se encontra na região norte do país.

Uma técnica de MD que responde bem ao problema de agrupamentos por similaridade é a técnica de Clusterização, apresentada na Seção 2.1.3. Foi então preparado um *dataset* formado apenas com os atributos *Fornecedor* e *UF*, sendo que *Fornecedor* é a empresa que participou da licitação, e *UF* é a UF do órgão que está licitando. Este *dataset* foi preparado para que a técnica de clusterização encontrasse os prováveis grupos de UF simbolizando mercados de licitações. A similaridade entre os grupos de UF nesse caso, seria obtida a partir dos fornecedores comuns às UFs do grupo. Desta forma, poderíamos

executar a técnica de Regras de Associação em cada grupo de UFs, já que teríamos em cada grupo um provável mercado de licitações, dividindo desta forma o espaço de soluções.

O teste foi executado nas 26612 instâncias da base, tendo como atributos Fornecedor e a UF onde participou da licitação, conforme apresentado na Figura 4.3. Foi encontrado um modelo de *clusters* contendo 10 regiões. Na Figura 4.1, pode-se notar que a maioria dos *clusters* encontrados tem como característica a proximidade geográfica, com exceção do *Cluster 1*, formado por duas subregiões separadas geograficamente.

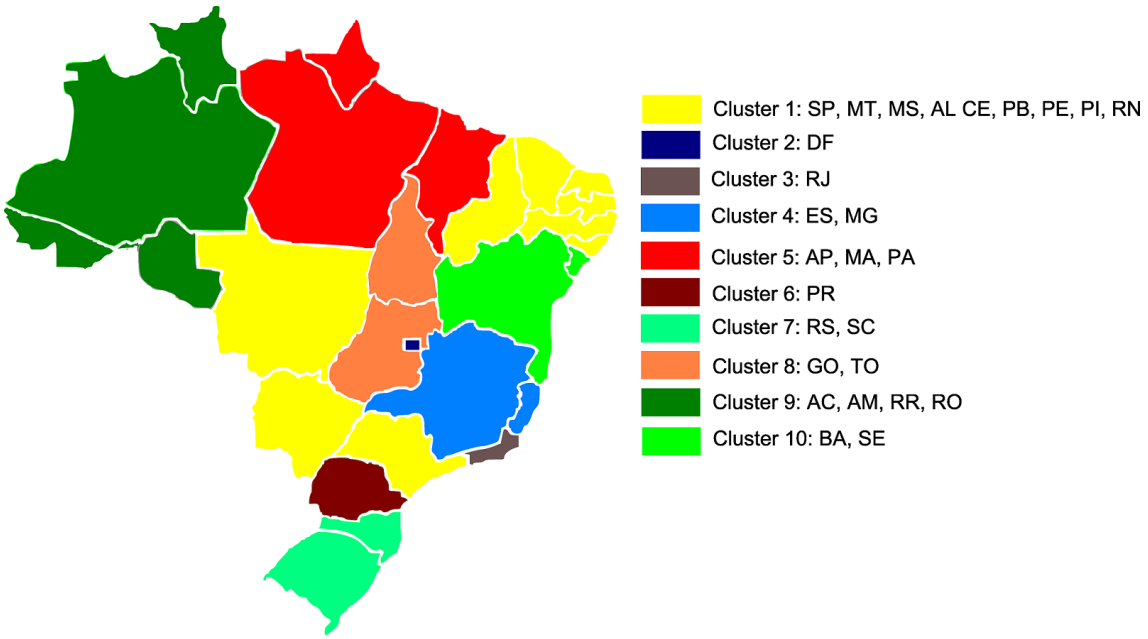


Figura 4.1: Classificação Preliminar de UF nos *Clusters*

Com as regiões definidas pelos grupos de UF encontrados na aplicação do algoritmo de clusterização *EM*, foi possível aplicar a técnica de Regras de Associação em cada *cluster*, na tentativa de identificar grupos de empresas associadas atuando especificamente nas novas regiões definidas. Além disso, com menos instâncias, foi possível configurar o valor de suporte mínimo para extrair regras cujo número de ocorrências fosse maior ou igual a nove, isto é, para que o grupo de fornecedores apontado por uma regra de associação tivesse atuado em pelo menos nove licitações, possibilitando assim a descoberta de novos grupos suspeitos para se analisar. Os resultados deste experimento podem ser vistos na Tabela 4.5.

4.1.3 Avaliação das Regras de Associação

Ao se comparar as Tabelas 4.4 e 4.5, pode-se notar a diferença entre o número de regras produzidas em ambos os experimentos. A diminuição do suporte mínimo no algoritmo acarreta uma maior descoberta de Regras de Associação. Mas nem todas as regras indicam propriamente um caso de provável cartel em licitações, podem indicar apenas coincidência ou mesmo frequentes atuações de empresas em licitações. Se a empresa participa de várias licitações e figura em vários grupos diferentes encontrados nas regras, é provável que seja apenas uma empresa cujo foco é o mercado de licitações públicas. Desta forma, para filtrar

Tabela 4.5: Execução do Apriori para *datasets* de *clusters*

Cluster	**Inst.**	**Atrib.**	**Sup.**	**Conf.**	**Regras**
1	787	614	2%	80%	851
2	211	164	4%	80%	1406
3	261	166	3%	80%	100
4	194	257	5%	80%	86
5	134	168	6%	80%	115
6	98	152	9%	80%	2848
7	270	196	4%	80%	1679
8	94	118	1%	80%	3
9	211	204	4%	80%	22
10	134	259	10%	80%	5869
Σ	2394	2298	-	-	12979

melhor as regras obtidas, definimos um método de avaliação com ajuda dos especialistas. A Equação 4.1 apresenta este método.

$$RQ = 100.\frac{V(F)}{Sup. \times Inst.} \tag{4.1}$$

Na Equação 4.1, *Sup.* é o valor do suporte da regra (em %), e *Inst.* é o número de instâncias do *dataset* ao qual a regra pertence. Foi inserida também na Equação 4.1 a função $V(F)$, que retorna o número de vezes que algum fornecedor do grupo apontado pela regra venceu as licitações, considerando as licitações em que todo o grupo participou. A função $V(F)$ terá como valor máximo de seu resultado o próprio suporte da regra avaliada multiplicado pelo número de instâncias do *dataset*. De forma mais simples, a Equação 4.1 retornará a probabilidade do grupo identificado na regra vencer as licitações de que participam, retornando um valor entre 0 e 100.

Normalmente, taxas altas de suporte e confiança determinam o quão boa uma Regra de Associação é. Suponha que uma Regra de Associação seja encontrada mostrando uma forte associação entre três fornecedores atuando nas licitações de um *dataset*. Isso poderia ser indício de um rodízio de licitações. No entanto, quando verificamos na base, quase nenhuma das vezes em que esses três fornecedores participaram juntos de licitações e algum deles conseguiu fechar um contrato com a Administração Pública. Portanto, suas participações nos mesmos processos de licitação foram uma mera coincidência. A confiança da regra não entrou na equação pelo fato de termos definido sempre um valor mínimo alto de confiança (70-80%), conforme apresentado nas Tabelas 4.4 e 4.5.

Para análise dos resultados preliminares, mensurados pela Equação 4.1, foram selecionadas as 10 melhores regras segundo a função de avaliação. As melhores regras obtidas apenas pela aplicação de Regras de Associação tiveram, na média, melhores números de ocorrências (suporte multiplicado pelo número de instâncias) quando comparadas com as melhores regras obtidas através da aplicação de Regras de Associação no espaço clusterizado; estas regras, entretanto, tiveram um aumento de cerca de 100% no valor de avaliação. O gráfico da Figura 4.2 mostra, para cada *cluster*, a avaliação obtida nas suas 10 melhores regras.

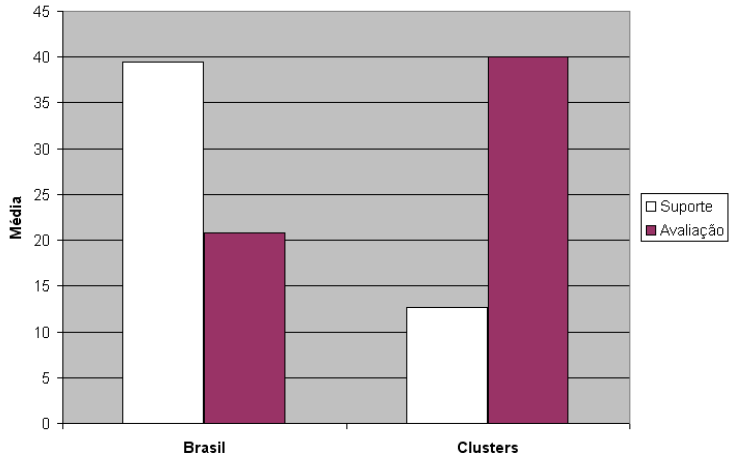


Figura 4.2: Média de Suporte e Avaliação das 10 melhores regras

Este resultado mostra que as melhores regras na nossa base, segundo a avaliação adotada, tende a aparecer quando o suporte é baixo, e quando se há uma melhor definição do espaço de soluções, neste caso, definido pelos *clusters* encontrados. Por isso as regras abrangendo o Brasil todo não foram tão boas quanto as encontradas em regiões do país.

Entre os modelos gerados a partir dos *clusters* (Tabela 4.5), as melhores regras foram obtidas no *Cluster* 6. A comparação entre as 10 melhores regras obtidas nesses modelos pode ser vista no gráfico da Figura 4.3.

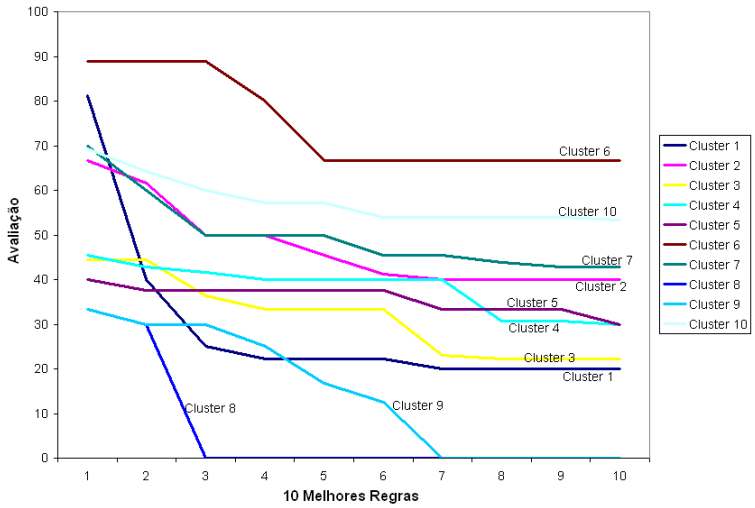


Figura 4.3: Comparação das 10 melhores regras dos Modelos de *Clusters*

Devido ao tempo gasto com tarefas manuais, tais como preparação dos dados, criação de *datasets*, adaptação dos resultados para avaliação, entre outros, não foi possível colher de forma exata o tempo total gasto na execução do experimento. Percebemos apenas que a execução total das tarefas levou dias de trabalho. No entanto, estas tarefas podem ser adaptadas para execução de agentes através de SMA. Os aspectos de distribuição, automatização de tarefas e execução paralela de algoritmos são adequados à integração

em SMA. Assim, é preciso propor um forma para melhorar o tempo de execução de tarefas, levando-se em conta que a repetição desse processo, de forma manua,l para outras bases de dados de licitações, é extremamente oneroso. Na próxima Seção apresentamos *Data Mining for Agents Framework* (*DMA Framework*), uma extensão da biblioteca de algoritmos de MD do Weka para ser utilizada em ambiente multiagente.

4.2 *DMA Framework*

Conforme citado no Apêndice A.1, o Weka, por ser de código aberto, permite a alteração de código fonte e incorporação de novas funcionalidades `http://www.cs.waikato.ac.nz/`.

Na abordagem de MD voltada para SMA, a que é apresentada na Seção 4.1, encontramos algumas dificuldades, especialmente no formato utilizado pela biblioteca do Weka 3.6.1 para apresentação dos modelos de MD encontrados. Para facilitar a integração dos algoritmos de MD com o SMA, alteramos alguns códigos da biblioteca do Weka e incluímos algumas outras funcionalidades, para facilitar principalmente a manipulação de resultados, criação de *datasets* e adaptação das estruturas de MD do Weka para as estruturas da ontologia definida no SMA.

O *DMA Framework* é uma extensão da biblioteca Java do Weka adaptada para ser utilizada em ambiente multiagentes. Foi desenvolvido na linguagem Java e é composto por três pacotes básicos, contendo um total de 14 classes. A estrutura do Diagrama de pacotes é mostrada na Figura 4.4.

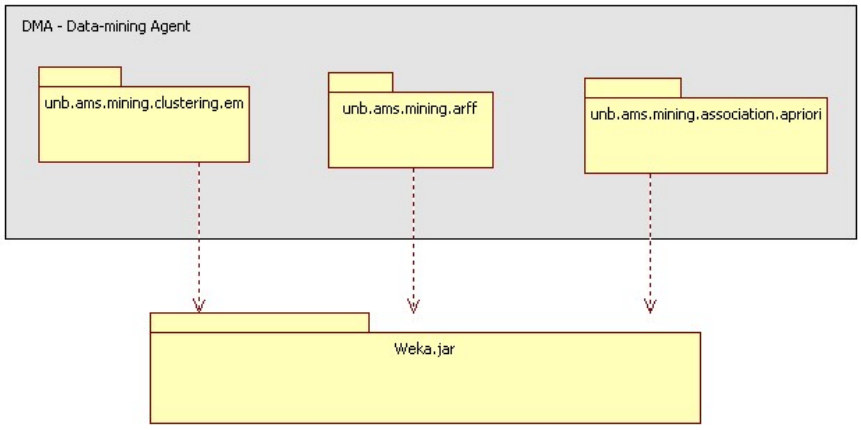


Figura 4.4: Diagrama de pacotes do DMA

O pacote *unb.ams.mining.arff* reúne três classes para criação de *datasets*, extraindo os dados de um banco de dados e adaptando-os ao formato exigido pela biblioteca do Weka. Para os nossos testes, desenvolvemos algoritmos para criação de instâncias no formato exigido para algoritmos de clusterização e algoritmos de regras de associação, seguindo a solução apresentada na Seção 4.1. Neste caso, o *DMA Framework* disponibiliza funcionalidades para transformação de dados de uma coleção em atributos, para criação de *datasets* compatíveis com a técnica de regras de associação. Através dessas funcionalidades

é possível converter os dados de uma coluna de tabela de um banco de dados em atributos para um *dataset* associativo.

O pacote *unb.ams.mining.association.apriori* reúne classes de apoio para regras de associação utilizando o algoritmo Apriori. O diagrama apresentado na Figura 4.5 mostra as classes destacando apenas alguns atributos e métodos presentes.

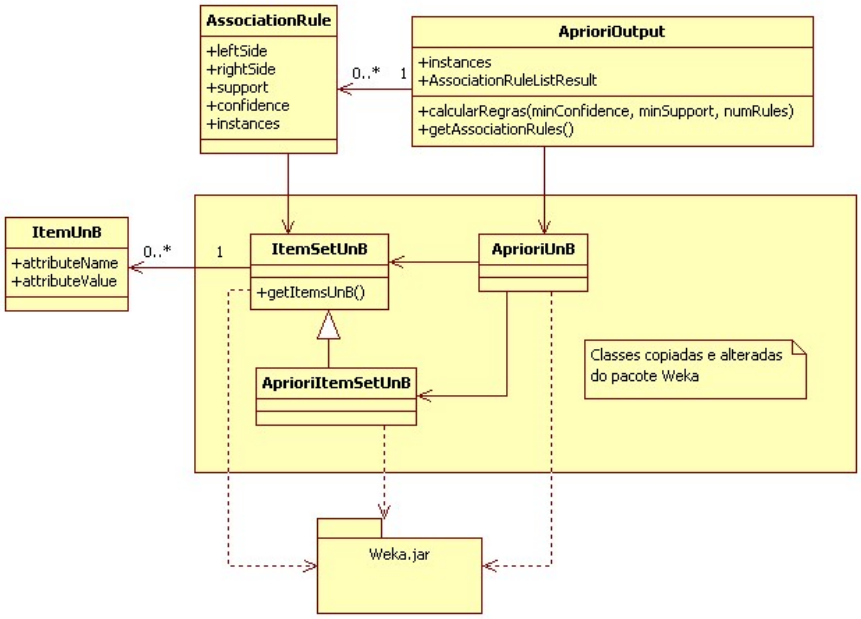


Figura 4.5: Diagrama de classes: regras de associação do DMA - algoritmo Apriori

A classe *AprioriUnB* é uma alteração da classe *Apriori* da biblioteca Weka, permitindo o acesso a algumas estruturas de dados no intuito de facilitar a manipulação dos modelos obtidos através da execução do algoritmo. A classe *AprioriOutput* é a classe que provê o serviço de mineração de dados, e retorna o resultado na estrutura definida pela classe *AssociationRule*.

O pacote *unb.ams.mining.clustering.em* reúne classes de apoio para execução do algoritmo de clusterização EM. O diagrama apresentado na Figura 4.6 mostra as classes destacando apenas alguns atributos e métodos presentes.

A classe *EMClassifier* provê o serviço de mineração de dados, através do algoritmo EM, e retorna o resultado na estrutura definida pela classe *EMCluster*. A estrutura criada suporta apenas clusterização com dois atributos, levando em conta o escopo de nosso experimento. No entanto, pode ser estendido sem muitas dificuldades para mais dimensões.

4.3 AGMI

As fases do processo de MD, apresentadas na Seção 2.1.1, requer muito trabalho, em especial a preparação dos dados e a avaliação do modelo encontrado. Além da preparação do *dataset* para execução da técnica de Clusterização, e de outro *dataset* para execução da técnica de Regras de Associação na base toda, para cada *cluster* encontrado foi necessário

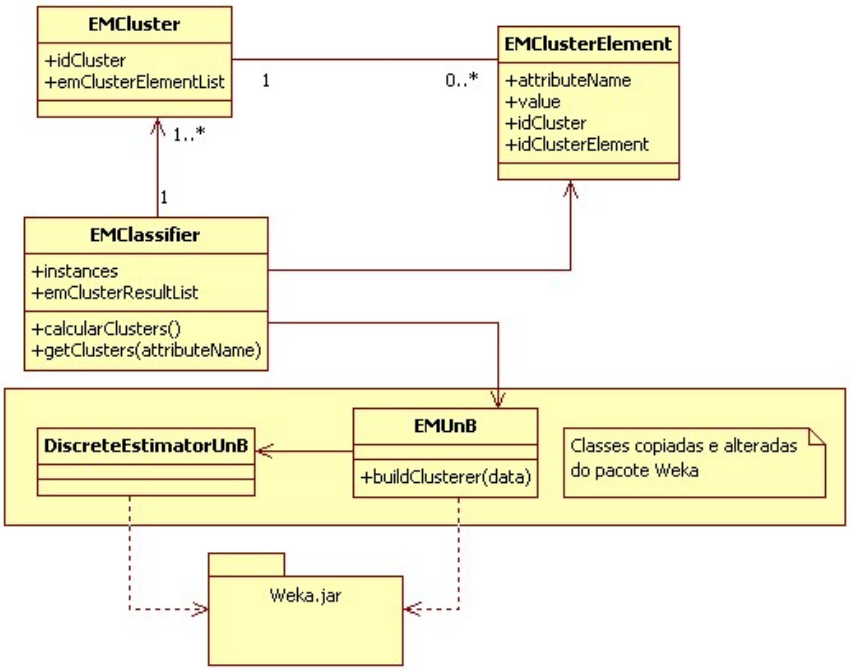


Figura 4.6: Diagrama de classes: técnica de clusterização do DMA - algoritmo EM

construir um novo *dataset* conforme a solução apresentada na Seção 4.1, e executar a técnica de Regras de Associação em cada um. E por fim, colher os resultados na ferramenta Weka para análise do conhecimento encontrado.

Como nossa abordagem requer baixo valor de suporte mínimo, o número de regras cresceu significativamente. Sendo assim, para análise das regras, era necessário extrair os resultados da ferramenta e adaptá-los a uma planilha eletrônica para melhor manuseá-los. No entanto, para utilização de uma abordagem dessa para diferentes bases de dados, o processo é demasiadamente oneroso e, consequentemente, impraticável. Por essa razão, foi criado o *DMA Framework*, introduzido na Seção 4.2.

Outro fato percebido nos testes com MD é a característica favorável de se adaptar as tarefas em ambiente distribuído. Levando em conta essa conveniência de se distribuir as tarefas, a execução de MD em ambiente distribuído, em especial no ambiente de SMA, possibilita, além de uma significativa redução de tempo de execução, a introdução de mecanismos para melhorar modelos encontrados, testar esses modelos, e apresentá-los de forma compatível com as necessidades do negócio.

A Figura 4.7 mostra AGMI, numa visão estrutural. AGMI opera com o agente Coordenador, o agente Avaliador de Regras, e vários times de MD. No modelo da figura, o Coordenador e o Avaliador de Regras aparecem no contêiner principal da plataforma JADE, por serem agentes essenciais no processo. No entanto, é completamente possível que esses agentes operem em outro contêiner. É através dos contêineres que a plataforma JADE possibilita a distribuição dos agentes em diferentes máquinas, dessa forma, os times de MD podem ser alocados em diversos contêineres do JADE, possibilitando a distribuição dos serviços entre *hosts* distintos. Para mais informações sobre a plataforma JADE, ver o Apêndice A.2.

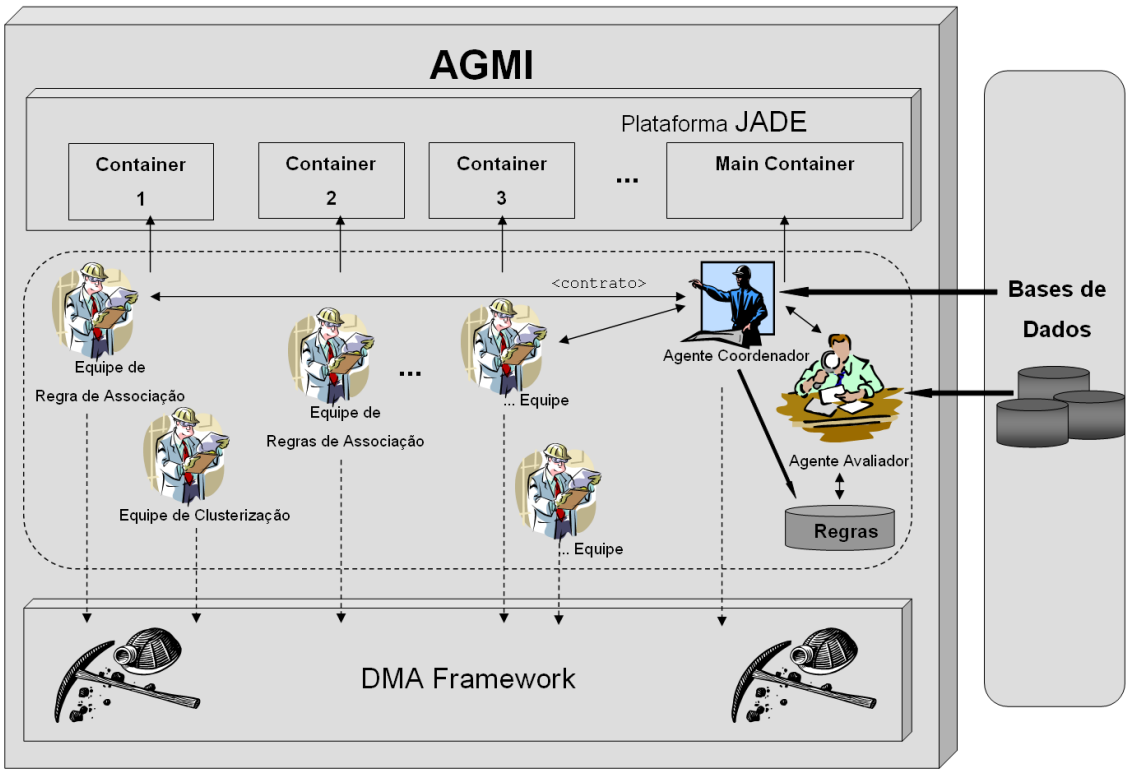


Figura 4.7: AGMI numa visão estrutural

Os times de MD provêm serviços de MD, cujos algoritmos são disponibilizados no *DMA Framework*. O Coordenador se relaciona com os times de MD através de contratos de serviços. Os agentes Avaliador de Regras e Coordenador são os únicos com permissão de leitura nas bases de dados. Além das bases de dados, o agente Avaliador de Regras administra uma base de regras. Nesta base de regras, é permitida a escrita por parte do Coordenador e do Avaliador, no entanto, a recuperação das regras para análises mais minunciosas fica a cargo apenas do agente Avaliador de Regras.

A Figura 4.8 apresenta uma visão mais detalhada das camadas da AGMI. Na camada superior, chamada de Estratégica, estão situados os agentes que possuem uma visão global do sistema. Esses agentes possuem objetivos que envolvem todo o sistema e elaboram estratégias e ações para alcançá-los. Eles decidem quais tarefas são necessárias; possuem uma visão mais abrangente do ambiente, conhecendo os dados, os resultados já produzidos, o conhecimento encontrado, os serviços pendentes, as equipes de trabalho, entre outros.

Na camada intermediária - a Tática - encontram-se os agentes chamados Supervisores de Equipes de Mineração. Esses agentes atuam cada um em sua área funcional, que no contexto de AGMI são os serviços de MD providos. As equipes são especializadas em um determinado tipo de técnica de MD, e o Supervisor é o responsável por negociar os serviços prestados, e gerenciar a execução dos mesmos administrado os recursos disponíveis em sua equipe. Podemos dizer que nesta camada, encontram-se as decisões de nível intermediário e especializado.

Na camada operacional encontram-se os agentes chamados Mineradores. Os agentes Mineradores compõem as equipes de MD e são gerenciados pelos seus Supervisores na execução dos algoritmos de MD. Nesta camada, cada agente executa um algoritmo de

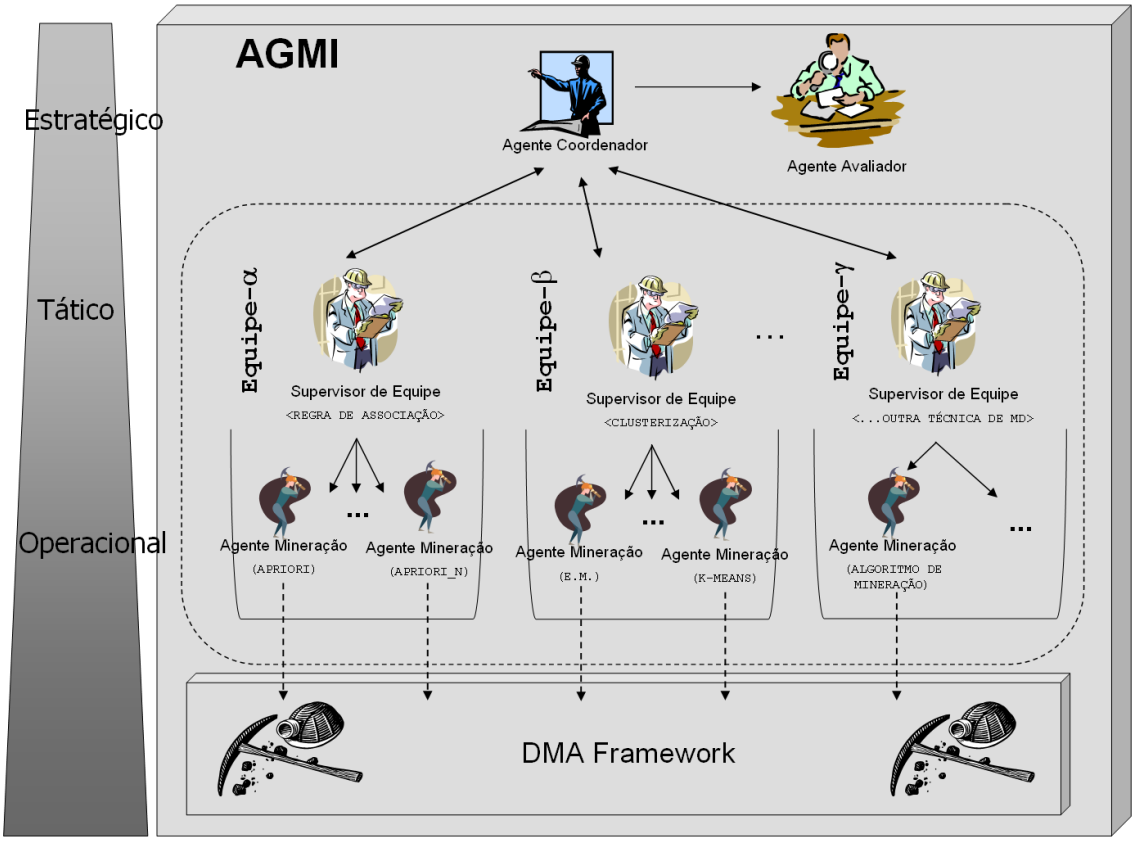


Figura 4.8: AGMI numa visão em camadas

MD compatível com a especialidade de sua equipe. Não há uma obrigação de todos terem algoritmos distintos entre si, podendo existir vários agentes mineradores executando algoritmos iguais na mesma equipe.

As equipes de MD mantêm uma estrutura de organização funcional. Na descrição encontrada em PMBOK (2010) acerca dessas organizações, nota-se que elas são hierárquicas e cada trabalhador tem o seu superior bem definido. O pessoal é agrupado por suas especialidades, no caso de AGMI, pelas técnicas de MD. No entanto, esta estrutura não se aplica no sistema como um todo pelo fato dos Supervisores não se remeterem ao Coordenador como um superior hierárquico, pois a relação, neste caso, é negocial, tratando-se de contratos de serviços, os quais detalharemos mais adiante.

4.3.1 Descrição dos Agentes de AGMI

Os agentes de AGMI tem como ambiente as bases de dados, a base de conhecimento (ou base de regras) e os *datasets* produzidos durante a execução das tarefas. Os agentes da camada Estratégica possuem uma visão mais abrangente desse ambiente. Já os Supervisores de equipes de MD têm uma visão mais restrita do ambiente, visualizando apenas os dados com os quais trabalharão e seus modelos de MD produzidos. Esses dados e modelos, no entanto, são apenas visualizados no momento da realização do contrato de serviço com o Coordenador. Na camada operacional, cada agente Minerador acessa apenas seu *dataset* para execução do algoritmo de MD, devolvendo o modelo encontrado.

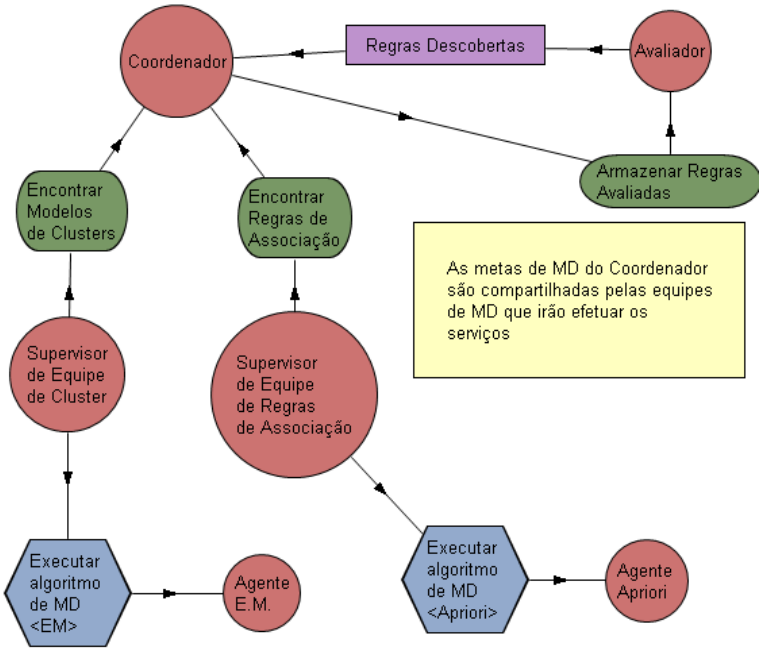


Figura 4.9: Diagrama de dependências gerado na fase de requisitos iniciais

Na Figura 4.9 podemos verificar numa visão de requisitos o diagrama de atores da Metodologia de Tropos, gerado pela Ferramenta TAOM4E, conforme apresentado na Seção 2.2.4. Neste diagrama pode ser observado de forma geral os atores (agentes) do processo e as dependências estratégias que os ligam na arquitetura. No diagrama, temos duas metas ligando o Coordenador a Supervisores de equipes de MD: encontrar modelos de *clusters* e encontrar regras de associação. Já entre o Coordenador e o Avaliador, as Regras Descobertas figuram como recurso compartilhado, e pertencente ao Coordenador.

A seguir, descreveremos as características gerais de cada tipo de agente presente no AGMI.

Coordenador

Agente da camada estratégica que controla todo o curso das atividades desenvolvidas no processo de descoberta de conhecimento. É um agente orientado a objetivos, com conhecimento das tarefas básicas de descoberta de conhecimento a serem executadas. Este agente é também capaz de negociar, a partir dos resultados das tarefas básicas, novas tarefas para atingir seus objetivos.

Na Figura 4.10, pode-se observar o diagrama do agente Coordenador. A meta não funcional (*softgoal*) é encontrar grupos suspeitos de praticar cartéis em licitações. Essa meta é decomposta em duas sub-metas envolvendo tarefas de MD, que por sua vez são atingidas através da realização de tarefas de negociação, generalizada pela tarefa "leiloar serviço".

Para realização da tarefa de leilão dos serviços necessários, este agente localiza as equipes de MD e envia solicitação de serviços para as equipes disponíveis. Quando um contrato é realizado entre o Coordenador e uma equipe de MD (através do Supervisor), o

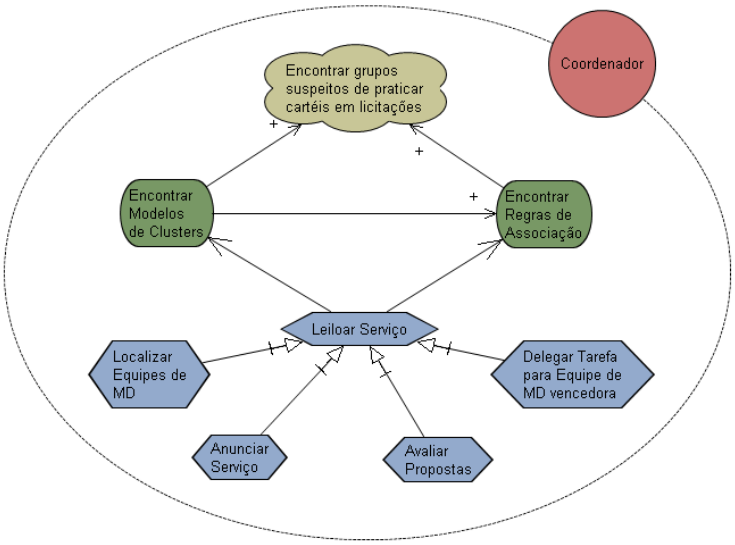


Figura 4.10: Diagrama do agente Coordenador gerado na fase de requisitos iniciais

Coordenador prepara os dados no formato necessário para realização da atividade de MD requerida e os repassa ao seu contratado.

Avaliador de Regras

Agente responsável pela pontuação e classificação do conhecimento encontrado durante o processo de MD. É também um agente orientado a objetivos. O diagrama de requisitos iniciais deste ator pode ser visto na Figura 4.11.

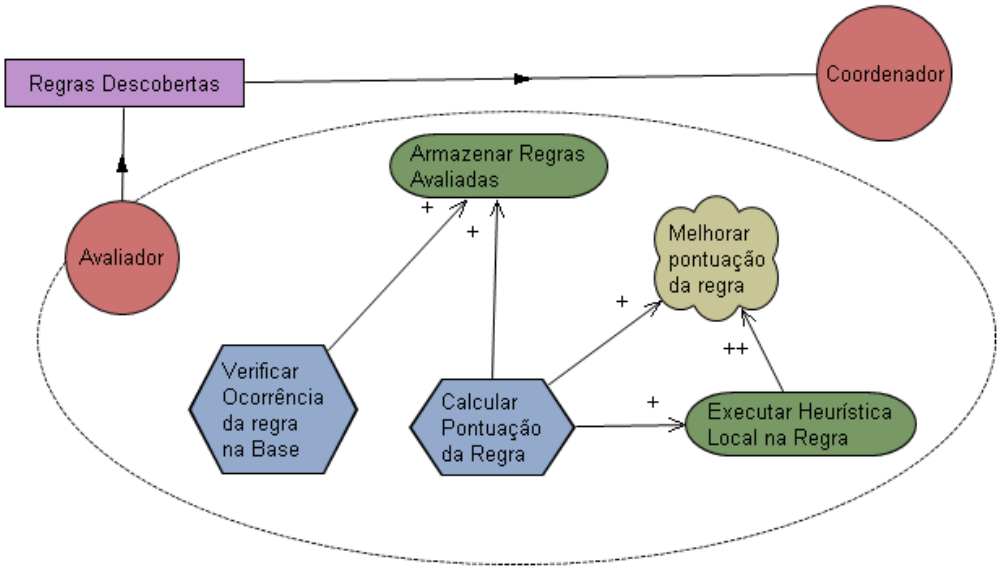


Figura 4.11: Diagrama do Avaliador gerado na fase de requisitos iniciais

O agente avaliador tem como meta inicial armazenar as regras avaliadas. Para isso, ele inicialmente julga as regras através de parâmetros de medição de cobertura, acurácia e

qualidade, mantendo as melhores regras e eliminando as regras inferiores. Nesse processo, verifica-se a ocorrência da regra na base para evitar duplicidade de regras.

Além de sua atuação básica mensurando o conhecimento encontrado, este agente pode gerar outras regras a partir daquelas armazenadas no banco de regras, aplicando uma heurística para aumento da qualidade das regras. As regras que se encaixam em determinado padrão previamente definido são aplicadas localmente e têm seu suporte e sua pontuação qualitativa recalculados, na tentativa de criar uma regra com aplicação local superior à regra original encontrada pelas equipes de MD.

Supervisor de Equipe de MD

Este agente gerencia uma equipe de especialistas numa determinada tarefa de MD, tal como Regras de Associação, Clusterização, Classificação e Seleção de Atributos. Cada Supervisor deve ter sob seu comando um ou mais agentes Mineradores. Sua função principal é negociar seu serviço com o Coordenador, enviando uma proposta de contrato, e no caso de sua proposta ser aceita, repassar a tarefa à equipe de agentes mineradores. Quando todos os Mineradores da equipe já estão trabalhando, e a equipe recebe uma proposta de contrato, o Supervisor é o responsável por recusar formalmente a proposta. Este agente também é o responsável por repassar o *dataset* recebido do Coordenador, em caso de contrato estabelecido, ao agente Minerador selecionado para executar o serviço contratado. Pelo fato de trabalhar no sentido de aproveitar melhor seus recursos e também de ser capaz de negociar seus serviços, este agente pode ser considerado um agente orientado a objetivos.

Agente de Mineração

É o agente operacional do AGMI. Este agente executa o algoritmo de MD nos *dataset* preparados pelo coordernador a fim de se extrair deles modelos ou padrões úteis. Normalmente são os agentes que consomem a maior parte dos recursos computacionais, considerando o processamento de grandes volumes de dados. O agente minerador é um agente reativo. Dependendo da técnica de MD executada pelo agente, pode ser adaptado para se tornar um agente reativo com estado, por exemplo, um agente de regras de associação. Neste caso, o estado armazenado pelo agente serve para reajustar os critérios de processamento, tais como suporte e confiança.

A Figura 4.12 apresenta um agente minerador executando o algoritmo Apriori e sua meta não funcional de execução do algoritmo com os recursos de máquina disponíveis. Para auxiliar nessa meta, é realizada uma tarefa para reajuste do suporte mínimo quando a memória disponível para execução do algoritmo já não for suficiente. Pode-se notar também no diagrama que o *dataset* utilizado pelo agente é um recurso compartilhado pelo Supervisor da Equipe de MD.

4.3.2 Ciclo de Execução do AGMI

Nesta Seção, será descrito o funcionamento da aplicação de AGMI. O modelo foi arquitetado para utilizar a plataforma JADE e o *DMA Framework*, assim, o ciclo de execução de AGMI aborda desde a inicialização dos agentes, até sua finalização após a apresentação do conhecimento descoberto.

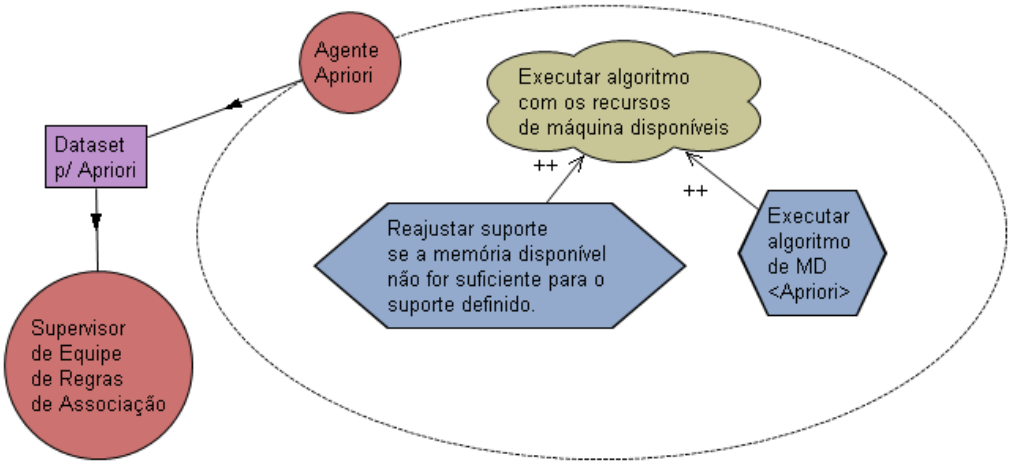


Figura 4.12: Diagrama do Agente de Mineração Apriori

Inicialização dos Agentes e Registro dos Serviços

AGMI foi modelado para que as equipes de MD possam atuar de forma autônoma, prestando, cada uma, os serviços de sua especialidade. Assim, é possível estender o modelo, inserindo outros agentes na camada estratégica, de modo que esses agentes sejam também capazes de contratar os serviços dos times.

Por causa disso, a execução de AGMI se inicia com o registro dos times de MD. Esse registro é feito no momento de inicialização dos agentes Supervisores. O registro é feito utilizando os serviços do agente *Directory Facilitator* (DF), responsável pela administração do serviço de Páginas Amarelas na plataforma JADE, mais detalhada no Apêndice A.2. A implementação de AGMI provê um catálogo de serviços passíveis de serem prestados na arquitetura para facilitar a localização destes por parte do agente Coordenador.

A Figura 4.13 apresenta um diagrama ilustrando esta interação, onde (1) é o momento de registro do serviço da equipe de MD, por parte do Supervisor, e (2) é o momento de localização dos prestadores de um determinado serviço, por parte do Coordenador.

A criação das equipes deve ser projetada pelo usuário. Para inicialização de uma equipe são definidos os seus *hosts* físicos, a quantidade de recurso que esta equipe poderá utilizar (memória), e a composição desta equipe, ou seja, número de agentes mineradores e seus algoritmos. AGMI pode ser estendida no sentido de ter uma configuração padrão, no entanto, devido às variações de tamanho de bases de dados, a experiência do especialista em MD parece ser uma melhor opção na configuração das equipes, escolhendo o número de times, hospedagem nas máquinas e composição dos times.

O Coordenador é programado para comandar, inicialmente, as tarefas principais de MD. Aplicando no contexto de rodízio de licitações, o agente Coordenador tem como meta inicial a realização das técnicas de Regras de Associação nos fornecedores, conforme solução apresentada na Seção 4.1, e clusterização na base de dados completa, levando em conta o atributo de UF para divisão do espaço geográfico de licitações. O Coordenador constrói então os *datasets*, utilizando as funcionalidades disponíveis no *DMA Framework*, e inicia o processo de contratação de serviço.

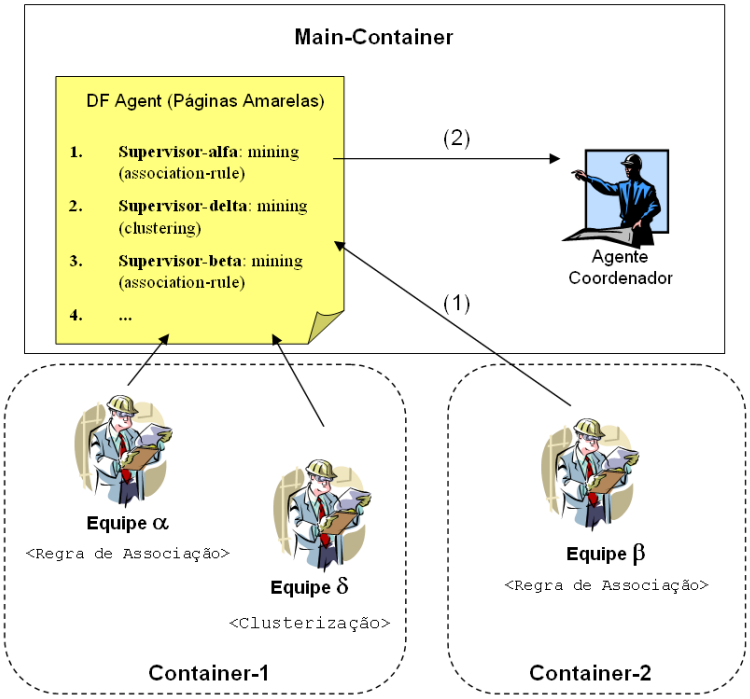


Figura 4.13: Registro dos serviços nas páginas amarelas

Contratação dos Serviços

O protocolo de interação utilizado no AGMI para negociação e contratação de serviços entre as equipes de MD e o Coordenador é o *Contract-Net*, apresentado na Seção 2.2.3. A Figura 4.14 ilustra a realização de uma negociação entre o agente Coordenador e duas Equipes de MD especializadas em Regras de Associação: Equipe *Alpha* e Equipe *Beta*. Essa negociação é feita através do protocolo *Contract-Net* e as equipes de MD são representadas pelos seus Supervisores.

Este protocolo realiza um tipo de licitação, onde os possíveis fornecedores de serviço, localizados pelas Páginas amarelas são comunicados sobre a contratação de serviço pelo contratador (agente Coordenador). Na plataforma FIPA, este momento é realizado por mensagens com a performativa *CPF*, acrônimo de *Call for Proposals*. No exemplo mostrado na Figura 4.14, nota-se que as mensagens capturadas nas linhas 30 e 31 são mensagens do tipo *CPF*.

Os agentes interessados em participar da licitação enviam suas propostas ao Coordenador, enquanto que as equipes que não estão interessadas em participar enviam uma mensagem de rejeição. As equipes podem rejeitar a participação nesse evento desde que já estejam no seu limite de trabalho, ou seja, sem agentes mineradores disponíveis para realização de novas tarefas. Caso contrário, o Supervisor da equipe sempre participa das licitações.

As propostas enviadas pelas equipes de AGMI são relativas a recursos disponíveis pela equipe. Cada equipe é configurada com uma quantidade de memória para trabalhar. Na medida em que os agentes mineradores são encarregados de uma tarefa, parte desse recurso é subtraído para fins de negociação. Caso uma equipe de três agentes mineradores seja configurada para operar com 3GB de memória e dois agentes estejam já realizando

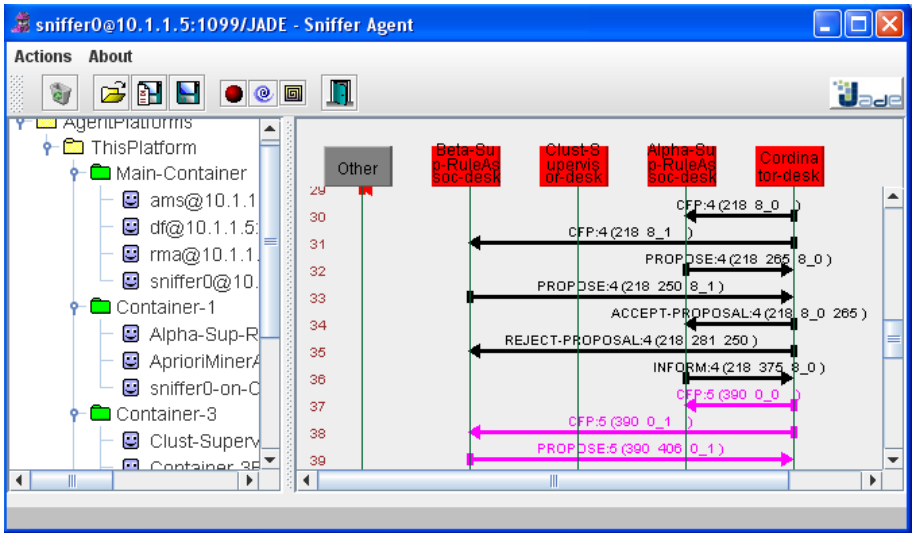


Figura 4.14: Exemplo da realização do protocolo Contract-Net capturado pelo agente *Sniffer* da plataforma JADE

suas tarefas, o Supervisor irá propor na negociação apenas 1GB para realização da tarefa. Este cálculo é formalizado através da Equação 4.2.

$$P = Mem_{max} - \frac{Mem_{max}}{|AgM|} \times |AgM_{ocupados}| \tag{4.2}$$

onde Mem_{max} é a quantidade máxima de memória disponível na JVM,
AgM é o conjunto de agentes de mineração da equipe e
$AgM_{ocupados}$ é o conjunto de agentes de mineração da equipe que estão trabalhando

O Coordenador, por sua vez, atua fazendo uso de uma estratégia gulosa, escolhendo sempre a proposta com maior recurso disponível. Na ilustração da Figura 4.14, as linhas 32 e 33 mostram as mensagens de propostas enviadas ao Coordenador, e as linhas 34 e 35 mostram as respostas do Coordenador, aceitando a proposta da Equipe *Alpha* e rejeitando a proposta da Equipe *Beta*, respectivamente.

O protocolo *Contract-Net* finaliza com o resultado enviado pelo agente que ganhou a negociação. No entanto, AGMI traz uma abordagem um pouco diferente por se tratar de realização de tarefas de MD, que normalmente consomem um tempo considerável de processamento. É enviada uma mensagem informando o compromisso da equipe de realizar a tarefa (linha 36 da 4.14) e o resultado é entregue de forma assíncrona, sendo controlado pelo Coordenador por meio do identificador da conversa.

Integração de Tarefas de MD

O Coordenador, assim que recebe os resultados encontrados nas tarefas de MD, decide se irá apenas armazenar na base de conhecimento ou, utilizá-los na integração com outras técnicas de MD. Na aplicação de AGMI, no contexto de Cartéis em Licitações, a técnica de clusterização é integrada com Regras de Associação para divisão do espaço de soluções. Desta forma, quando o Coordenador recebe o modelo encontrado de *clusters* geográficos,

ele gera para cada *clusters* um *dataset* correspondente, para que seja aplicada de forma regional a técnica de regras de associação, fazendo, em seguida, a contratação dos serviços.

Quando o espaço é dividido, os *datasets* gerados são consequentemente menores, possibilitando a redução do valor de suporte mínimo na execução da tarefa de Regras de Associação. Por causa disso, aumenta-se a probabilidade de encontrar regras com de maior qualidade.

Avaliação do Conhecimento Encontrado

Já foi mencionado que o agente Avaliador de Regras controla a pontuação e classificação do conhecimento encontrado no ciclo de execução de AGMI. Assim que novos conhecimentos são armazenados na base, o Avaliador é notificado pelo Coordenador. O Avaliador verifica se o novo conhecimento armazenado é passível de avaliação ou não - isso vai depender da implementação da função de avaliação no agente. Na aplicação de AGMI no problema de Cartéis em Licitações, por exemplo, não faz sentido, a princípio, a avaliação de um modelo de *clusters* encontrado através da Função 4.1. Para realização desta atividade, é possível também manter à parte uma base de conhecimentos avaliados. Neste caso, o Coordenador acessaria apenas a primeira base e o Avaliador, a segunda.

O agente Avaliador também é responsável por não permitir estruturas de conhecimento que sejam redundantes na base. No caso de Regras de Associação, é muito comum as regras terem a mesma semântica num determinado domínio. Tomando-se o exemplo de cartéis em licitações, novamente, uma Regra de Associação $A, B, C \rightarrow D$ com taxa de confiança de 100% tem a mesma semântica da regra $A, D \rightarrow C, D$ e $B, C \rightarrow A, D$, por exemplo, se essas também tiverem a taxa de confiança de 100%.

No momento da análise das regras, o Avaliador decide se a regra analisada é passível de ser melhorada ou não, através da aplicação de heurísticas. Quando todas as tarefas forem completadas, o Coordenador comunica o agente Avaliador, para que apresente a base de conhecimento classificada e filtrada pelos parâmetros estabelecidos no SMA, tais como: qualidade mínima desejada, número máximo de regras, entre outros.

Heurística para Melhoramento de Regras de Associação

Na aplicação de AGMI para solução do problema de cartéis, foi desenvolvida uma heurística para melhoramento de Regras de Associação encontras. Cada regra encontrada aplica-se ou a uma região específica de um *cluster* geográfico, formado normalmente por algumas UFs ou, à região formada por todas as UFs. No entanto, é possível que um grupo de empresas que pratiquem cartel tenha mais sucesso em uma região ainda mais específica, reunindo o maior número de contratos em uma única UF, por exemplo.

Durante a avaliação das regras, o agente Avaliador obtém informações de quantas vitórias o grupo apontado pela regra possui em cada UF do *cluster* aonde aquela regra é aplicada. Partindo dessas informações, o Avaliador classifica esses números selecionando a UF aonde ocorreu o maior número de vitórias por parte do grupo apontado pela regra. Se esse número encontrado é superior ao limite mínimo estabelecido no agente Avaliador, ele aplica aquela regra na UF selecionada, calculando o suporte local e o novo valor da qualidade baseada na Função 4.1, verificando, em seguida, se a regra aplicada localmente é melhor ou pior que a original. Caso seja melhor, a original é substituída pela regra nova.

Para que a regra seja substituída, é necessário também que a nova regra tenha os valores de suporte e confiança maiores ou iguais aos limites estabelecidos no sistema.

Comunicação

AGMI foi projetado para trabalhar na plataforma JADE, que é uma implementação da especificação FIPA para SMA. A comunicação entre os agentes nesta plataforma é feita através da linguagem FIPA-ACL, já apresentada na Seção 2.2.3.

Segundo Bellifemine et al. (2007), na plataforma JADE, encontram-se três importantes tipos de elementos usados para construção de ontologias. Esses elementos são derivados da linguagem FIPA-ACL:

- Conceitos - expressões que indicam entidades com uma estrutura complexa que pode ser definida em termos de *slots*. Exemplo: (Pessoa :nome João :idade 33). Conceitos tipicamente não fazem sentido se usados diretamente como conteúdo de uma mensagem FIPA-ACL. Geralmente eles são referenciados dentro de um predicado ou dentro de outro conceito. Exemplo: (Livro :título "A Abolição do Homem" :autor (Pessoa :nome "C.S. Lewis")).
- Predicados - são expressões que dizem algo sobre o estado do mundo e podem ser verdadeiras ou falsas. Exemplo: (Trabalha-para (Pessoa :nome João) (Empresa :nome OxLab)), significando que João trabalha para a empresa OxLab. Predicados podem ser significativamente usados para mensagens FIPA-ACL do tipo INFORM e QUERY-IF.
- Ações de Agente - um conceito especial que indica ações que são executadas por algum agente. Exemplo: (Venda (Livro :título "Ortodoxia") (Pessoa :nome José)). É útil tratar as ações de agentes separadamente uma vez que, diferentemente dos conceitos usuais, as ações podem fazer mais sentido em certos tipos de *ACLMessages*, tais como as do tipo *REQUEST*. Atos comunicativos (istó é, mensagens FIPA-ACL) são eles mesmos ações de agentes.

Para melhor clareza e organização nas trocas de informações, foi definida uma ontologia com alguns conceitos básicos para comunicação dos agentes em AGMI. No entanto, esta ontologia pode ser estendida para se adaptar em diferentes aplicações de AGMI. Optamos por utilizar o *plugin OntologyBeanGenerator* da ferramenta *Protégé*, apresentada na Seção 2.2.3. A escolha se deu pelo fato *OntologyBeanGenerator* ser um *plugin* para integração da ferramenta com o ambiente JADE. O modelo pode ser visto na Figura 4.15.

Os conceitos da ontologia relativos às requisições de ações de MD devem ser uma especialização da ação principal *DoMiningTask*. No mesmo sentido, os diferentes tipos de resultados de tarefas de MD devem se especializar do conceito principal de *MiningResult*. Já as estruturas usadas para representação do conhecimento, isto é, modelos de MD, regras, *clusters*, entre outros, devem herdar o conceito *StructuredKnowledge*.

4.4 Protótipo

Foi implementado um protótipo de AGMI voltado para solução do problema de cartéis em licitações. Para desenvolvimento do SMA, foi utilizada a API do JADE. O fato de utilizar

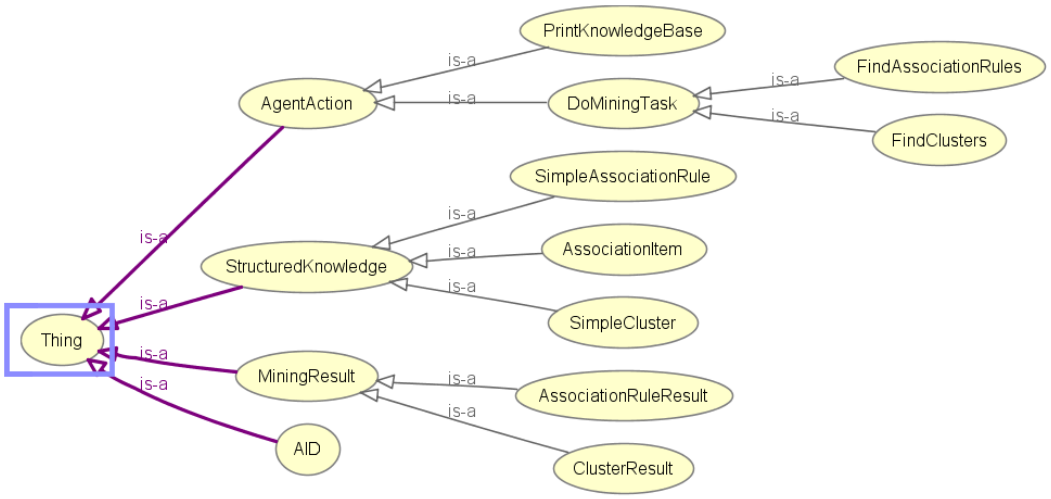


Figura 4.15: Modelo da Ontologia utilizada em AGMI

JADE como plataforma de SMA e *DMA Framework* para realização de tarefas de MD, faz-se necessário a utilização da linguagem Java para desenvolvimento do projeto. Para desenvolvimento do protótipo, utilizamos como plataforma de desenvolvimento a ferramenta *Eclipse*, bastante utilizada para desenvolvimento de aplicações Java. O servidor de banco de dados utilizado nos testes foi o *Microsoft SQLServer 2005*.

A preparação do ambiente, inicialização e execução dos agentes requer a instalação do *middleware* JADE, que pode ser baixado gratuitamente no site `http://jade.tilab.com`. Para inicialização de um agente na plataforma, é necessário que o projeto seja compilado e sua estrutura de arquivos binários seja copiada para o diretório raiz da plataforma. Além disso, as bibliotecas utilizadas (arquivos .jar) devem também estar localizadas no diretório *lib*, situado no diretório raiz da plataforma. A Seção A.2 apresenta de forma geral como se implementa um agente utilizando o *framework* JADE.

4.4.1 Visão Geral do Protótipo

O protótipo foi implementado utilizando uma organização em camadas. A Figura 4.16 mostra a organização dos pacotes no sistemas. A camada estratégica mostrada na Figura 4.8 é implementada no pacote *agent.management*. Já os agentes das camadas táticas e operacionais, que são os Supervisores de equipes e os agentes de mineração, são implementados em pacotes dentro de *agent.teams*. Estes agentes dependem dos serviços providos pelo *DMA Framework* e utilizam a ontologia para comunicação com os outros agentes. A dependência do *DMA Framework* por parte dos agentes do pacote *agent.management* é relativa ao serviço de preparação de *datasets* fornecido pelo *framework* e utilizado pelo agente Coordenador.

JADE provê em sua biblioteca uma estrutura de suporte à implementação de ontologias para comunicação entre agentes. A superclasse utilizada para construção de ontologia

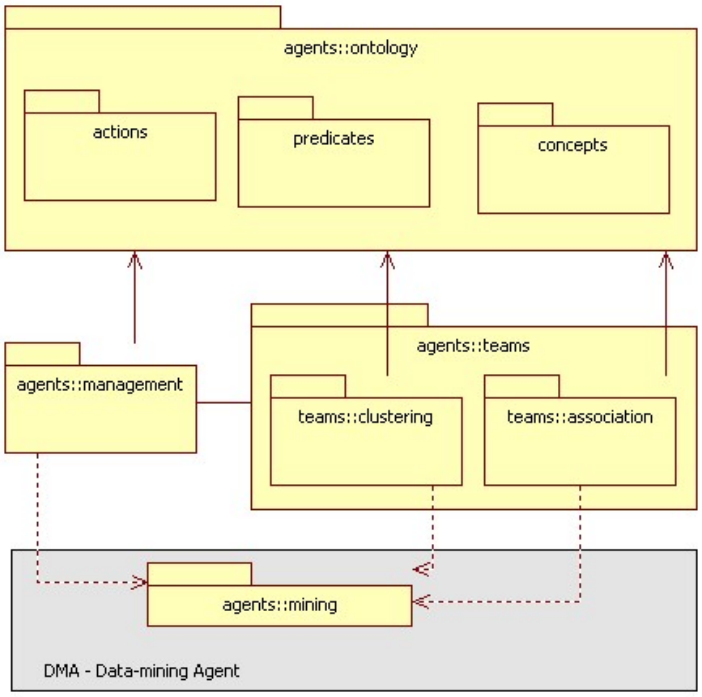


Figura 4.16: Diagrama de pacotes do protótipo

no JADE é *jade.content.onto.Ontology*. Uma extensão da classe *Ontology*, que permite a adição direta das classes a partir dos seus respectivos *JavaBeans*, é a classe *BeanOntology*. Para construção de uma ontologia utilizando *BeanOntology* no JADE, é necessário estender essa classe, definir as informações necessárias para criação da ontologia, tais como codec utilizado, nome da ontologia, e por fim, registrar as classes *JavaBeans* que implementam os *ObjectSchemas*, que podem ser conceitos, predicados e ações da ontologia. Desta forma, para se implementar os *ObjectSchemas*, a classe deve implementar ums dessas três superclasses: *Concept*, *AgentAction* ou *Predicate*.

No nosso protótipo, implementamos a ontologia para comunicação dos agentes conforme definido no modelo taxonômico, ilustrado na Figura 4.15. No protótipo, demos o nome de *AGMI-ontology*. Note que para as mensagens FIPA-ACL do tipo *INFORM* só permitem em seu conteúdo elementos de ontologia do tipo predicado, o que faz sentido uma vez que uma informação deve ser uma estrutura em que seja possível atribuir um valor de verdade. Assim, para reportagem do resultado de uma tarefa de mineração, nossa ontologia usa o predicado *ResultsOf*, que relaciona uma tarefa de MD e um resultado obtido. A implementação desse elemento e suas relações é ilustrado no diagrama de classes mostrado na Figura 4.17.

4.4.2 Camada Estratégica

A camada estratégica de AGMI é composta pelos agentes Coordenador e Avaliador (Figura 4.8).

O Coordenador utiliza objetos das classes *CoordinatorContractNetInitiatorBehaviour* e *CoordinatorListeningBehaviour* para definição de seus comportamentos. Dentro do

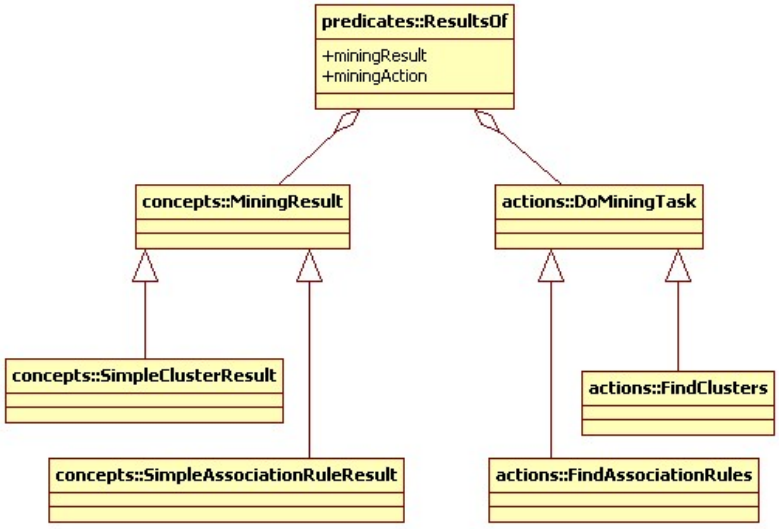


Figura 4.17: Diagrama de clases ilustrando a composição do predicado *ResultsOf*

Coordenador é também definido outro comportamento para contratação sob-demanda de serviços de regras de associação baseados em *clusters* encontrados. O diagrama de classes do agente Coordenador é mostrado na Figura 4.18.

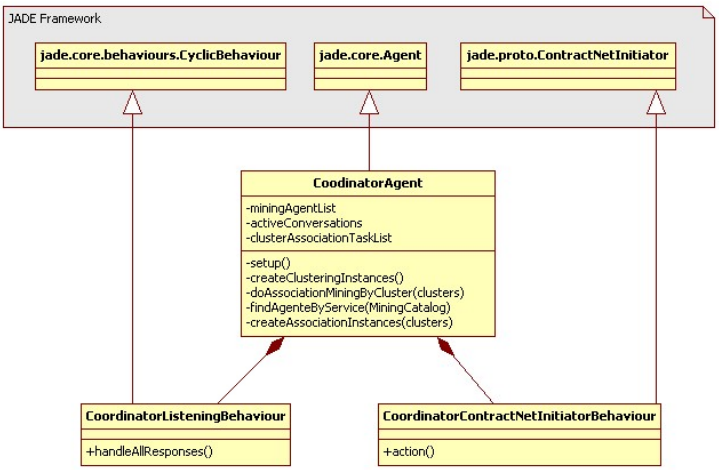


Figura 4.18: Diagrama de classes do agente Coordenador

A classe *CoordinatorContractNetInitiatorBehaviour* provê o comportamento de negociação de serviços utilizando o *Contract-Net* implementado em JADE. Essa classe estende a superclasse *jade.proto.ContractNetInitiator* que provê a implementação do papel inicial do protocolo (Figura 2.9), isto é, envio das mensagens *Call for Proposals* aos interessados. O iniciador solicita propostas de outros agentes, enviando na mensagem *Call for Proposals*, a especificação da ação a ser executada e as condições de sua execução.

A classe *CoordinatorListeningBehaviour* provê o comportamento de escuta de mensagens relativas a resultados de tarefas de MD do Coordenador. Assim, nesta classe,

são aplicados como filtros de escuta a ontologia *AGMI-ontology*, o codificador escolhido (*LEAPCodec*), e a performativa *INFORM*. Além disso, o comportamento foi configurado para ignorar as mensagens recebidas sob o protocolo *Contract-Net*. Isto porque o comportamento *CoordinatorContractNetInitiatorBehaviour* também atua recebendo mensagens. Sendo assim, para que as mensagens destinadas às ações de negociação não sejam consumidas pelo comportamento *CoordinatorListeningBehaviour*, as mensagens sob o protocolo *Contract-Net* são ignoradas. JADE possibilita este tipo de controle de recebimento de mensagens graças ao recurso chamado *MessageTemplate* disponibilizado no *framework*. Este tipo de configuração é exemplificado no trecho de código a seguir:

```
1 ...
  //definição do template
3
    MessageTemplate template = MessageTemplate.and(MessageTemplate.
        MatchOntology(AGMIOntology.getInstance().getName()),
5        MessageTemplate.MatchLanguage(AGMIOntology.getCodec().getName()));

7   template = MessageTemplate.and(template, MessageTemplate.
        MatchPerformative(ACLMessage.INFORM));

9   template = MessageTemplate.and(template,
        MessageTemplate.not( MessageTemplate.MatchProtocol(FIPANames.
            InteractionProtocol.FIPA_CONTRACT_NET))
11      );

13 //recebimento da mensagem com filtro
  ACLMessage msg = myAgent.receive(template);
15 ...
```

Após a realização da atividade de clusterização pela equipe responsável, o Coordenador é notificado do resultado e precisa realizar atividades de regras de associação em *datasets* gerados a partir dos *clusters* encontrados. Desta forma, para cada *cluster* encontrado, deve ser realizada uma atividade de regras de associação. Sendo assim, no agente Coordenador é implementado um método que provê contratação de serviços de regras de associação sob-demanda. Este comportamento difere do *CoordinatorContractNetInitiatorBehaviour*, porque as primeiras contratações (um serviço de regras de associação e um serviço de clusterização) são realizadas na inicialização do sistema, e tem-se a garantia de que pelo menos uma equipe de cada tipo de técnica de MD existirá no sistema e estará disponível. Após o recebimento dos *clusters*, o Coordenador não sabe quantas equipes de MD estão disponíveis, nem tampouco quantos agentes de mineração existem nas equipes. Desta forma, o Coordenador deve realizar quantas tentativas de contratação sejam necessárias para que todos os serviços sejam realizados.

Para realização desta tarefa, é adicionado no Coordenador um novo comportamento que, de tempos em tempos, inicia várias tentativas de contratação de serviços de forma sequencial, utilizando um *SequentialBehaviour*, tendo como sub-comportamentos instâncias de *CoordinatorContractNetInitiatorBehaviour*. Por exemplo, se foi descoberto 12 *clusters*, o agente dispara inicialmente 12 tentativas de contratação de serviços para MD. Se nessa primeira rodada foram contratadas três equipes, na próxima rodada o agente irá disparar nove tentativas de contratação de serviço, até que todos os serviços sejam delegados.

A Figura 4.19 mostra através do diagrama de atividades como este comportamento funciona. Estão envolvidos nesta tarefa três tipos de comportamentos. Um mais externo,

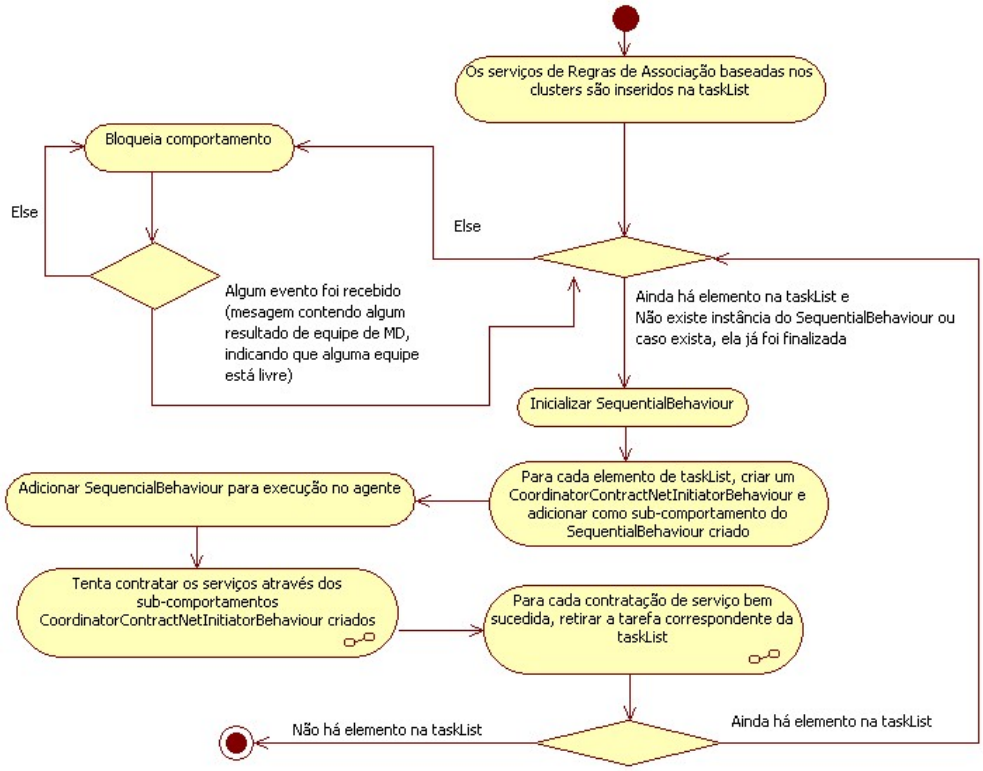


Figura 4.19: Diagrama de atividades da contratação de serviços de regras de associação sob-demanda

que é adicionado para que a atividade seja realizada concorrentemente com as outras atividades do agente, tal como a atividade de escuta, para recebimento de resultados de equipes de MD. O outro comportamento utilizado é um comportamento sequencial, para agrupamento e execução das tentativas de contratação de serviços de forma sequencial. E, por fim, os comportamentos de contratação de serviço, feitos através de instâncias do *CoordinatorContractNetInitiatorBehaviour*. Para execução das atividades mostradas na Figura 4.19, é necessária a criação de uma lista (*taskList*), para armazenamento das tarefas pendentes. Inicialmente, essa lista é configurada com um número de tarefas igual ao número de *clusters* encontrados. Na medida em que os serviços vão sendo delegados, as tarefas vão sendo removidas da lista. Além dessa lista, é necessário um objeto do *SequentialBehaviour*, o qual será instanciado em cada iteração, com um número de sub-comportamentos igual ao número de tarefas pendentes na *taskList*.

O agente Avaliador é implementado no protótipo, gerenciando a base de conhecimentos avaliados. O agente tem apenas um comportamento cíclico para recepção de mensagens. As mensagens esperadas pelo agente são: notificações de novas regras encontradas e mensagem para geração de relatório das regras avaliadas. Esse relatório é gerado apenas após todas as regras encontradas serem avaliadas pelo agente. Após a impressão do relatório, o agente é finalizado, encerrando o trabalho. No protótipo, esse relatório é gerado no formato *csv* (*comma-separated values*). A Figura 4.20 mostra as classes mais importantes envolvidas na construção do agente.

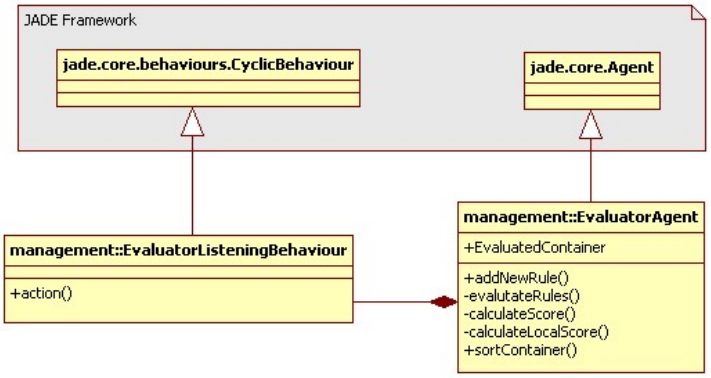


Figura 4.20: Diagrama de classes do agente Avaliador

Quando são recebidas mensagens de notificação de novas regras encontradas, o Avaliador inicia o processo de avaliação das regras, checando inicialmente se existem regras semelhantes na base de conhecimento. Caso as regras sejam redundantes, elas são descartadas. Caso contrário, o avaliador executa o método *evaluateRules* para pontuar as regras. Caso a regra que está sendo avaliada apontar um grupo de empresas com um número de vitórias nas licitações em que todo o grupo participou maior ou igual a um número definido no sistema (no protótipo o número escolhido foram 4 vitórias), o avaliador executa a heurística para tentar melhorar a regra aplicando-a localmente. Isso é feito através do método *calculateLocalScore*. Caso a heurística produza uma regra com suporte mínimo maior ou igual ao limite definido no sistema, uma regra nova é gerada semelhante à original, no entanto, com local de aplicação diferente da primeira. A implementação do protótipo permite que no relatório seja listada a regra de associação com suporte, número de vitórias e pontuação global e local, como mostrado na Tabela 4.6.

Tabela 4.6: Informações das regras apresentadas no relatório do agente Avaliador

B, C, D (12) → K sup. (12), conf. (1,00)	
Informação	**Valor**
Cluster	2
Suporte	12
Vitórias	5
Pontuação	41
Local	BA
Suporte(L)	10
Vitórias(L)	5
Pontuação(L)	50
Melhoria (%)	22,0

4.4.3 Equipes de MD

As equipes de MD são formadas por um agente Supervisor de equipe e um ou mais agentes de mineração. O agente Supervisor de equipe é um agente da camada tática e os agentes de mineração da camada operacional, como já descrito na Seção 4.3.1.

No nosso protótipo, foram implementados dois tipos de equipes de MD: equipe de regras de associação e equipe de clusterização. A implementação dos agentes Supervisores são similares, uma vez que utilizam a mesma lógica de operação, variando apenas a formulação de propostas a serem enviadas no momento de negociação de tarefas. Outra característica que muda de Supervisor para Supervisor é o tipo de serviço registrado no momento de inicialização. Cada Supervisor deve cadastrar o serviço que provê no momento de inicialização. Os serviços foram catalogados no protótipo por meio da classe *MiningServiceCatalog*. O trecho de código a seguir mostra como o agente Supervisor de Regras de Associação faz o registro do serviço prestado por sua equipe.

```
1  private void serviceRegistration() throws FIPAException{
     DFAgentDescription dfAgentDesc = new DFAgentDescription();
3    ServiceDescription aRService = new ServiceDescription();
     aRService.setName(MiningServicesCatalog.ASSOCIATION_RULES.getName());
5    aRService.setType(MiningServicesCatalog.ASSOCIATION_RULES.getType());
     dfAgentDesc.setName(getAID());
7    dfAgentDesc.addServices(aRService);
     DFService.register(this, dfAgentDesc);
9  }
```

Na inicialização de um Supervisor de equipe de MD, deve ser passado como parâmetro o número de agentes de mineração que a equipe terá. Além disso, é possível definir a quantidade de memória dedicada à execução do agente, através da configuração dos argumentos de configuração de memória na inicialização da JVM. Os argumentos configurados são *-Xms* e *-Xmx*, para configuração do tamanho inicial e final da pilha (*heap*) Java, respectivamente.

O Supervisor de equipe precisa ser capaz de interagir com o Coordenador do SMA através do protocolo *Contract-Net*. No entanto, sua tarefa não é a de iniciar a interação, e sim, responder à mensagem *call for proposal* enviada pelo Coordenador. Desta forma, cada Supervisor de equipe de MD deve implementar uma classe, estendendo a superclasse *jade.proto.ContractNetResponder*, que provê os atributos e métodos para a atividade de resposta ao protocolo *Contract-Net*. A Figura 4.21 apresenta o diagrama de classes contendo a classe do agente Supervisor de Clusterização e a classe dos agentes de mineração *EM*, além das classes de comportamentos e suas superclasses.

Como já mencionado na Seção 4.3.2, o protocolo *Contract-Net* finaliza com o resultado enviado pelo agente que ganhou a negociação. No entanto, na implementação de AGMI, a finalização do protocolo se dá através de uma mensagem informando o compromisso da equipe de realizar a tarefa contratada e o resultado é entregue de forma assíncrona, sendo controlado pelo Coordenador, por meio do identificador da conversa. Essa abordagem foi definida por causa da metodologia do *framework* JADE implementar um agente.

Na Seção A.2, foi mencionado que cada agente JADE é implementado numa única *thread*, sendo assim, a concorrência provida pela plataforma aos diversos comportamentos do agente é uma concorrência artificial, não dispondo de mecanismos preemptivos para escalonamento dos comportamentos. Assim, cada comportamento tem o seu método *action* executado completamente, sendo verificado no método *done()* se aquele comportamento

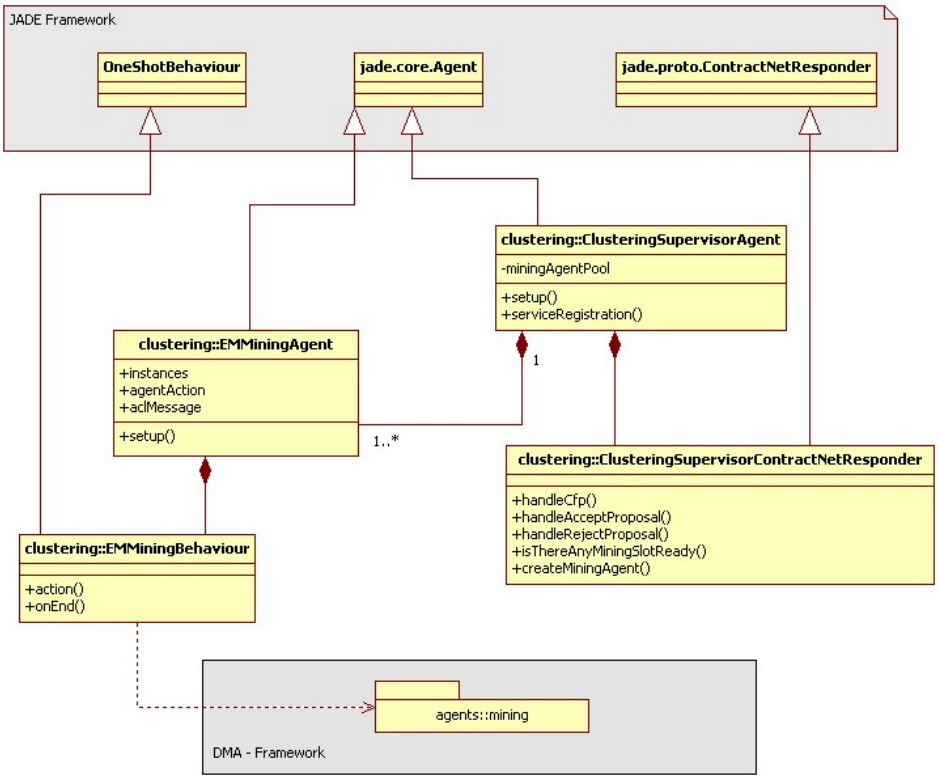


Figura 4.21: Diagrama de classes da equipe de MD de Clusterização

já foi terminado. Caso tenha terminado, o comportamento sai da fila de comportamentos do agente, caso contrário, ele é colocado no fim da fila para que o próximo inicie o processamento do método *action*. Este fluxo é mostrado na Figura A.5.

Algoritmos de MD normalmente consomem muito tempo de execução. Desta forma, se o agente Supervisor esperar a execução de seu comportamento (no caso, o *ContractNetResponder*) até que o agente de mineração termine de processar o algoritmo enviando o resultado, todos os comportamentos do Supervisor estarão bloqueados, não sendo permitido a ele responder a outras propostas de contrato, podendo assim sub-utilizar os recursos da equipe.

Na Figura 4.21 pode-se observar a classe *ClusteringSupervisorContractNetResponder* que estende a superclasse *ContractNetResponder* e implementa o comportamento para execução do protocolo *Contract-Net* pelo Supervisor da Equipe de Clusterização. Nesta classe, são reescrito os métodos *handleCfp()* para tratamento das chamadas para propostas do protocolo, além dos métodos *handleAcceptProposal()* e *handleRejectProposal()* para lidar, respectivamente, com a aceitação e a rejeição da proposta por parte do Coordenador. No método *handleCfp()*, o Supervisor verifica se há agentes disponíveis na equipe para realização da tarefa. Existindo, o Supervisor envia a proposta para o Coordenador, de acordo com a Equação 4.2, aguardando em seguida o resultado. O Supervisor recusará prestar serviço ao Coordenador quando o número máximo de agentes trabalhando for atingido. Isso é checado através do método *isThereAnyMiningSlotAvailable*.

O mecanismo de criação e gerenciamento dos agentes de mineração das equipes de MD foi implementado no protótipo de forma que os agentes de mineração, uma vez definida

a quantidade na inicialização do Supervisor, sejam criados no momento da realização da tarefa de MD, sendo finalizado imediatamente após a entrega dos resultados encontrados. Essa abordagem foi utilizada para evitar o acúmulo de objetos em memória na execução de algoritmos de MD. Essa preocupação se deve muito mais ao algoritmo de regras de associação que consome maior quantidade de memória em sua execução. Assim, toda vez que uma tarefa de MD é negociada na equipe, um agente de mineração é criado para executá-la, sendo morto após o término de execução da tarefa. Por isso o comportamento dos agentes de mineração estende a classe *OneShotBehaviour*, cuja característica é realizar o método *action()* apenas uma vez, encerrando em seguida o comportamento. O atributo *aclMessagem*, mostrado na classe *EMMiningAgent* na Figura 4.21 é a mensagem enviada pelo Coordenador com a tarefa designada. Esta mensagem carrega o identificador da conversa realizada no contrato, e o endereço do agente Coordenador. Isto torna possível o agente de mineração enviar o resultado diretamente ao Coordenador, após finalizar o processamento. Essa quebra de hierarquia na implementação tem finalidade de evitar o *overhead* gerado por excessivas trocas de mensagens.

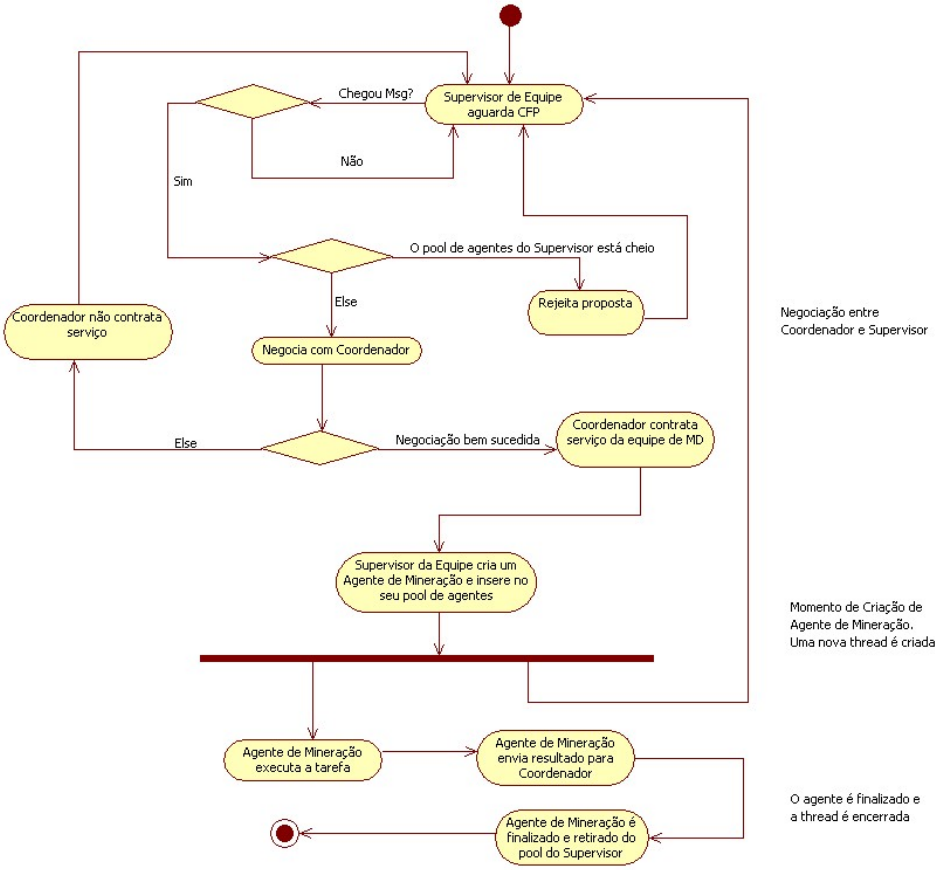


Figura 4.22: Diagrama de atividades mostrando a criação de agentes mineradores pelo Supervisor de um Equipe

A Figura 4.22 mostra, através do diagrama de atividades, o processo de negociação e criação de agentes pelo Supervisor de uma Equipe de MD. Note que o tamanho do *pool* de agentes de mineração de um Supervisor de Equipe de MD é definido no processo de

inicialização do mesmo. Quando o *pool* de agentes está completo, significa que o Supervisor já está com sua carga máxima de trabalho, rejeitando assim qualquer solicitação de negociação por parte do Coordenador. Esta checagem é feita através do método booleano *isThereAnyMiningSlotAvailable()* (Figura 4.21). Este método checa se o *pool* de agentes mineradores do Supervisor está completo ou não. Caso não esteja completo, o Supervisor pode enviar uma proposta ao Coordenador, participando, assim, da disputa pelo serviço em questão. Caso o Supervisor da Equipe ganhe o contrato, ele então cria um agente de Mineração para execução do trabalho, inserindo ele no seu *pool*. O agente criado, pelo fato de ser uma nova *thread* Java, é executado de forma concorrente ao Supervisor, não interrompendo assim os outros processos de negociação daquela Equipe. Este processo é feito através do método *createMiningAgent()*, mostrado também na Figura 4.21. Este método tem uma lógica semelhante na implementação dos dois tipos de equipes desenvolvidas no protótipo (clusterização e regras de associação). A seguir, é apresentado o código deste método na classe do agente Supervisor de Regras de Associação (*AssociationSupervisorAgent*).

```
1  private void createMinerAgent(FindAssociationRules fa, ACLMessage msg,
       int index) {
     //dataset sendo extraído da mensagem
3    Instances i=null;
     try {
5      i = (Instances) msg.getContentObject();
     } catch (UnreadableException e1) {
7      e1.printStackTrace();
     }
9
     Object[] parameters = new Object[] { i, fa, msg };
11   int num = 0;

13   //caso seja um dataset de regras de associação baseado em clusters
     //utilizado na nomenclatura do agente
15   if(fa.getSimpleCluster() != null){
       num = fa.getSimpleCluster().getNumCluster()+1;
17   }

19   try {
       String agentName = "AprioriMiningAgent("+num+")_"+myAgent.
           getContainerController().getContainerName();
21     aprioriMiningAgentPool[index] = myAgent.getContainerController()
           .createNewAgent(agentName, AprioriMinerAgent.class.getName(),
               parameters);
23     aprioriMiningAgentPool[index].start();
     } catch (StaleProxyException e) {
25     e.printStackTrace();
     } catch (ControllerException e) {
27     e.printStackTrace();
     }
29 }
```

4.4.4 Mecanismo Autônomo de Reajuste de Suporte para Agentes de Regras de Associação

Como apresentado na Seção 2.1.4, algoritmo de regras de associação tende a consumir grande quantidade de memória no processamento de dado, sempre que o suporte mínimo é configurado com valor baixo. No entanto, o consumo de memória depende também do conjunto de dados que está sendo processado. Dois conjuntos de dados de mesmas proporções podem apresentar diferentes consumos de memória, sendo processados com as mesmas taxas de suporte e confiança. Isto decorre da seleção de itens de dados frequentes na base para construção das regras, o que pode variar de um conjunto de dados para outro.

Pelo fato de estarmos lidando com regras de baixa frequência, no problema dos cartéis em licitações, o valor de suporte mínimo configurado baixo pode ocasionar em recorrentes problemas de estouro de memória, dependendo do tamanho do conjunto de dados que se está processando. Assim, num processo normal de MD, o algoritmo deve ser ajustado até que se consiga um valor compatível com os recursos disponíveis.

Na implementação de AGMI, inserimos no agente de Regras de Associação um mecanismo autônomo para contornar o problema de estouro de memória no processamento dos dados. O protótipo normalmente trabalha com um valor mínimo de suporte configurado para processamento do algoritmo de regras de associação em todos os *datasets* gerados. Desta forma, sempre que o agente de regras de associação inicia o processamento e o nível de memória utilizado atinge o máximo alocado para a *thread* do agente, o valor de suporte mínimo é reajustado para um valor maior, iniciando assim um novo processamento daquele *dataset*. Isto possibilita a adaptação do protótipo em qualquer base de dados seja qual for o seu tamanho, e seja qual for o recurso disponível para processamento dos dados. Obviamente que quanto mais recursos estiverem disponíveis, mais regras poderão ser encontradas, pois o agente poderá trabalhar com valores baixos de suporte. A seguir, pode-se notar o método *runApriori* do protótipo, onde é implementado esse mecanismo.

```
1  private AprioriOutput runApriori(double support, double confidence, int
       maxRules){
     System.gc();
3    AprioriOutput ao = new AprioriOutput(instances);
     try{
5      ao.calcularRegras(support, confidence, maxRules);
     }catch(java.lang.OutOfMemoryError om){
7      if(support == (double)1.0){
         ao = null;
9      }else if(support*2 > (double)1.0){
         ao = runApriori(1.0, confidence, maxRules);
11     }else{
         ao = runApriori(support*2, confidence, maxRules);
13     }
     }
15   return ao;
   }
```

A característica de agente reativo com estado é observada no momento em que o agente não consegue terminar o processamento por falta de memória e, então, baseado no valor anterior do suporte adotado, decide reiniciar o processamento adotando um novo valor de suporte - que no caso do código apresentado é o dobro do valor anterior. Note

que este comportamento é feito de forma recursiva finalizando quando o processamento se completa ou quando o valor de suporte atinge 1.0, que é o valor para localização de regras com ocorrência em 100% das instâncias do *dataset*.

No próximo capítulo, apresentaremos os experimentos realizados com AGMI para processamento de dados de licitação obtidos do Sistema ComprasNet. Serão apresentados experimentos utilizando uma implementação integrando agentes de software e posteriormente os utilizando AGMI com diferentes equipes de MD e autonomias de agentes para melhoria de regras e da performance do sistema.

Capítulo 5

Experimentações e Análise dos Resultados

Neste capítulo nós apresentaremos os experimentos realizados desde a integração inicial de agentes de mineração, até a introdução de autonomia no agente Avaliador do protótipo de AGMI. O objetivo é relatar a evolução da pesquisa através dos diversos experimentos realizados. Desta forma, os resultados da utilização da integração inicial de SMA e MD foram comparados com a utilização da abordagem de AGMI, que apresentou um resultado superior nos aspectos de desempenho, autonomia dos agentes e melhoria na qualidade das regras.

5.1 Integração Inicial de Agentes de Mineração

Para os experimentos iniciais com o *DMA Framework* e o JADE, nós desenvolvemos três diferentes agentes: agente de mineração, o coordenador e o avaliador. O *dataset* usado nesse experimento foi o mesmo apresentado na Tabela 4.3 da Seção 4.1.1.

A maioria dos algoritmos de aprendizagem de máquina, como classificadores e clusterizadores são suceptíveis a ordenação dos dados (Weka, 2010). No Weka, quando se usa o algoritmo *EM* sem configurar o número de *clusters* desejados, é executada uma validação cruzada para determinar automaticamente o número de *clusters* para o modelo, da seguinte forma:

1. o número de *clusters* é configurado em 1
2. o conjunto de treinamento é dividido randomicamente em 10 *folds*
3. *EM* é executado 10 vezes usando os 10 *folds* do modo usual de validação cruzada
4. o logaritmo da verossimilhança é medido sob todos os 10 resultados
5. se o logaritmo da verossimilhança aumentou, o número de *clusters* é aumentado em 1, e o programa continua no passo 2.

Técnica de avaliação de modelos, tais como a Validação Cruzada de 10 Dobras, ou *10-fold cross-validation*, pode minimizar os possíveis erros de classificação. No entanto, pelo fato de *EM* classificar os elementos nos *clusters* através de distribuições probabilísticas,

a ordenação dos dados pode gerar modelos diferentes dependendo do *dataset* e de outros fatores tais como a semente randômica usada para inicialização do algoritmo. Desta forma, a menos que se execute o *10-fold cross-validation* 10 vezes e seja feita a média dos resultados, é provável que o algoritmo encontre modelos diferentes no mesmo *dataset*, levando em conta a ordenação dos dados (Tan et al., 2005; Weka, 2010).

Como utilizamos o algoritmo *EM* para dividir melhor o espaço de soluções, decidimos testar uma nova configuração de *clusters*, diferente da apresentada na Seção 4.1.2, a fim de explorar as possibilidades de obter novas regras em regiões geográficas. Para isso, os dados foram ordenados de forma diferente a fim de que obtivéssemos um segundo modelo de *clusters*.

Foram então realizados dois experimentos nos moldes daqueles apresentados na Seção 4.1.1. O primeiro experimento utilizou apenas a técnica de Regras de Associação, e o segundo integrou Regras de Associação com Clusterização. No entanto, nestes experimentos foram utilizados agentes para automatização das tarefas de MD, além do *DMA Framework*. O objetivo foi de mensurar o tempo gasto no processo através da utilização dos agentes para automatização das tarefas, possibilitando a geração de novas regras com a utilização da nova ordenação dos dados para descoberta de um novo modelo de *clusters*. As Seções 5.1 e 5.2 apresentam a realização destes dois experimentos.

5.1.1 Experimento com Regras de Associação

Neste experimento foram usados três agentes: Coordenador, Avaliador e de Regras de Associação, o qual utilizou o algoritmo *Apriori* como descrito na Seção 4.1.1. O valor de suporte mínimo configurado foi 0.9% e o valor mínimo de confiança foi 90% considerando o *dataset* com todas as licitações da base de dados (2701 instâncias). Este valor de suporte foi definido empiricamente baseado na capacidade de memória das máquinas disponíveis para teste, e permitindo a procura de regras com no mínimo 24 ocorrências no *dataset*. Para medir a qualidade das regras encontradas, nós implementamos no agente Avaliador a Equação 4.1, conforme apresentado na Seção 4.1.3. A distribuição dos agentes é apresentada na Tabela 5.1, e os resultados do experimento na Tabela 5.2.

Tabela 5.1: Distribuição dos Agentes

Host A	Intel Core 2 2.40 GHz, 2.00 GB RAM	
Host B	Intel Pentium Dual 1.86 GHz, 2.00 GB RAM	
	Host A	**Host B**
Agente Coordenador	X	
Agente Avaliador	X	
Agente de Regras de Associação		X

Ressaltamos que o tempo de execução apresentado na Tabela 5.2 é relativo ao tempo gasto em todo o processo de MD, desde a preparação do *dataset*, até a avaliação das regras encontradas. O processo inicia com o Coordenador preparando o *dataset* para o agente de Regras de Associação. Este *dataset* é preparado utilizando a estratégia apresentada na

Tabela 5.2: Resultados da utilização dos agentes.

Informações	**Números**
Regras Encontradas	128
Tempo de Execução	29'
Média de RQ (100 melhores regras)	16.56
Média de Suporte (100 melhores regras)	30.98
Média de RQ (10 melhores regras)	33.00
Média de Suporte (10 melhores regras)	26.90

Seção 4.1. Na sequência, o agente de Regras de Associação executa o algoritmo *Apriori* enviando os resultados ao Coordenador. Finalmente, o agente Avaliador analisa as regras, gravando o resultado em arquivo.

Na Tabela 5.2, pode-se notar também a diferença entre a média de suporte das 100 melhores regras e das 10 melhores regras. Isso confirma mais uma vez que valores altos de suporte não garantem regras de boa qualidade, no caso de cartéis em licitações. No entanto, a execução de Regras de Associação em toda a base de dados limita a redução do suporte e deixa o espaço de soluções muito esparsso. Por causa disso, integramos no processo a técnica de Clusterização para divisão do espaço geográfico em regiões. Assim, para cada região descoberta no modelo de *clusters*, um *dataset* de Regras de Associação é gerado para localização de grupos de fornecedores associados especificamente na região definida pelo *cluster*, possibilitando-nos trabalhar com um valor de suporte menor para obtenção de melhores regras.

Neste sentido, na Seção 5.1.2 será apresentado um experimento incluindo um agente de Clusterização no processo para divisão do espaço de soluções.

5.1.2 Experimento integrando Regras de Associação e Clusterização

Quatro agentes foram usados neste experimento: Coordenador, Avaliador, agente de Regras de Associação e agente de Clusterização. O agente de Regras de Associação executou o algoritmo *Apriori* e o agente de Clusterização executou o algoritmo *EM*, providos pelo framework Weka e adaptados no *DMA Framework*. No caso do agente de Regras de Associação, foi mantida a configuração de 0.9% para suporte e 90% para confiança, na execução do algoritmo do *dataset* principal, isto é, levando em conta todas as licitações e todos os fornecedores (2701 instâncias). Para as novas regiões definidas pelos *clusters* encontrados, nós configuramos o agente de Regras de Associação para executar o algoritmo com o suporte proporcional ao tamanho dos *datasets* gerados, de forma que o algoritmo fizesse buscas por regras de associação com no mínimo nove ocorrências no *dataset* examinado.

O processo completo, incluindo preparação de dados, execução de algoritmos de MD, avaliação de regras foi executado de forma automática pelos agentes. Além disso, os agentes de mineração (Clusterização e Regras de Associação) executaram seus algoritmos em paralelo. A distribuição dos agentes é apresentada na Tabela 5.3.

Nosso objetivo principal com esse experimento foi encontrar regras de associação que indiquem grupos suspeitos de praticar cartel em licitações. Além disso, a técnica de

Tabela 5.3: Distribuição dos Agentes.

Host A	Intel Core 2 2.40 GHz, 2.00 GB RAM
Host B	Intel Pentium Dual 1.86 GHz, 2.00 GB RAM

	Host A	**Host B**
Agente Coordenador	X	
Agente Avaliador	X	
Agente de Regras de Associação		X
Agente de Clusterização	X	

clusterização, dividiu o espaço para facilitar a busca regional por regras de associação, possibilitando perceber as regiões de licitações no país aonde as empresas normalmente participam. Com uma nova ordenação da base de dados, foi possível encontrar um segundo modelo de *clusters* conforme apresentado na Figura 5.1. Neste modelo, 7 regiões foram definidas, diferentes das 10 regiões encontradas no primeiro modelo de *clusters*, mostrado na Figura 4.1 da Seção 4.1.2. Apesar da configuração diferente, é interessante notar que os dois modelos de *clusters* apresentam similaridades entre as regiões encontradas. Quase todas levam em conta a proximidade geográfica e o compartilhamento de fronteiras entre os estados. Apenas o caso curioso do *Cluster* 1 da Figura 5.1, que apresenta na mesma região estados do nordeste e do centro-sul, com estados separados geograficamente.

Os resultados deste experimento são apresentados na Tabela 5.4. Note que os valores de qualidade das 10 melhores regras cresceram mais de 150% em relação à Tabela 5.2. Isso foi possível devido à utilização das regiões formadas pelos *clusters*. Se nós compararmos a média das melhores 100 regras dos dois experimentos (Tabelas 5.2 e 5.4), este experimento apresentou uma média 4 vezes maior que a média das 100 melhores regras do Primeiro Experimento.

Com a adição do agente de Clusterização nos nossos testes, o espaço de busca foi dividido, possibilitando a redução ainda maior do suporte mínimo, gerando, assim, um maior número de regras e com melhor qualidade.

Tabela 5.4: Resultados do experimento com integração de agentes de MD

Informações	**Números**
Regras Encontradas	6150
Tempo de Execução	75'
Média de RQ (100 melhores regras)	69.71
Média de Suporte (100 melhores regras)	9.78
Média de RQ (10 melhores regras)	89.70
Média de Suporte (10 melhores regras)	9.40

A reordenação dos dados feita pelos agentes no processo trouxe uma configuração de *cluster* diferente que possibilitou o aumento da qualidade das regras encontradas, comparando com as regras encontradas através da aplicação de MD sem agentes, apresentada

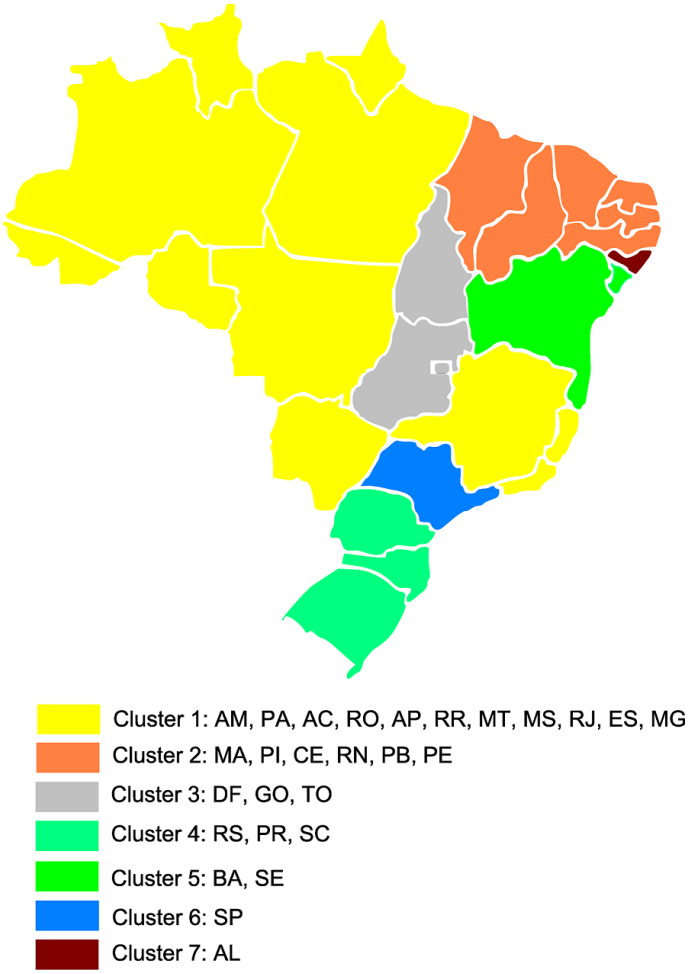


Figura 5.1: Segundo modelo de *clusters*, obtido através de uma nova ordenação dos dados

na Seção 4.1.3. Com o uso de agentes de MD, a qualidade das regras aumentou de 40% (Figura 4.2) para 89.7% (Tabela 5.4), considerando as 10 melhores regras.

Outro ganho significativo foi em relação ao tempo de execução, considerando os dias de trabalho gastos na execução das tarefas de MD citados na Seção 4.1.3. Neste experimento, o tempo de realização de todas as tarefas de MD foi de 75 minutos (Tabela 5.4). Esta redução está diretamente relacionada ao uso de agentes para automatização dos processos de preparação de dados, distribuição das tarefas de coordenação, execução dos algoritmos e avaliação das regras encontradas.

Comparando com o tempo de execução do primeiro experimento (29'), este segundo experimento teve um tempo de execução $2,6$ vezes maior (75'). Isto se deve à integração do agente de Clusterização. Este agente retornou como resultado um modelo composto por sete *clusters*. Assim, as tarefas de regras de associação foram aumentadas consideravelmente. No primeiro experimento, foi executada apenas uma tarefa de Regras de Associação, considerando toda a base de dados (2701 instâncias). Neste segundo experimento, além da tarefa de Regras de Associação em toda a base ser executada, outras sete tarefas de Regras de Associação relativas às regiões encontradas no modelo de *clusters* são também realizadas pelo agente, uma vez que para cada *cluster* do modelo, um novo *dataset* correspondente é criado para realização da tarefa de Regras de Associação.

Comparando os resultados das Tabelas 5.2 e 5.4, apesar do tempo ter aumentado cerca de 2,6 vezes, a qualidade das 10 melhores regras aumentou cerca de 2,7 vezes, provando o potencial de integração de MD e SMA. Na Seção 5.2, apresentaremos os experimentos realizados com o protótipo da ferramenta de AGMI.

5.2 AGMI

Nesta Seção, apresentaremos a evolução dos experimentos com o objetivo de reduzir o tempo de execução dos algoritmos, através do uso da abordagem distribuída de nossa arquitetura, e também de melhorar a qualidade das regras encontradas, através da utilização de heurística como estratégia de autonomia para o agente Avaliador.

Na Seção 5.2.1, são apresentados dois experimentos comparando os tempos gastos entre os processamentos com 1 e 2 equipes de regras de associação respectivamente. Na Seção 5.2.2 são apresentados os testes utilizando AGMI com o agente Avaliador aplicando heurística para melhoria de regras locais (autonomia do agente). Foram realizados experimentos com diferentes configurações de equipes de MD, além de um experimento fazendo uso de duas equipes de clusterização para integração dos dois modelos de *clusters* encontrados (Figuras 4.1 e 5.1), utilizando-os no mesmo processo de descoberta de conhecimento.

5.2.1 Experimento com Equipes de MD

AGMI utiliza equipes de MD responsáveis por prover serviços de uma determinada técnica de MD. Na Seção 5.1.2, apresentamos um experimento sem a presença das equipes de MD, como é apresentado em AGMI. Para fins de comparação com o experimento apresentado na Seção 5.1.2, configuramos as equipes de Regras de Associação e de Clusterização para operar com apenas um agente de MD, ou seja, cada equipe composta apenas pelo Supervisor da equipe e um agente de MD. A distribuição dos agentes e dos times de MD é mostrada na Tabela 5.5.

Tabela 5.5: Distribuição dos agentes entre os *hosts*

	Host A	**Host B**
Agente Coordenador	X	
Agente Avaliador	X	
Equipe de Regras de Associação α-Team	X	
Equipe de Regras de Associação β-Team		X
Equipe de Clusterização	X	

A Tabela 5.6 apresenta o tempo de execução do Teste 1. Para fins de comparação, colocamos na mesma tabela o tempo gasto no experimento apresentado na Seção 5.1.2 (75'). Levando em conta o número de agentes mineradores, ambos os experimentos são

similares, pois possuem apenas um agente de Regras de Associação e um agente de Clusterização. Além da presença do Supervisor negociando serviços com o Coordenador neste experimento, a outra diferença é a presença do agente Avaliador operando independentemente do Coordenador, uma vez que este agente se encontra na camada estratégica da arquitetura. Esta atuação do agente Avaliador de forma independente possibilitou a execução paralela das tarefas do Coordenador e do Avaliador. A diferença de tempo do Teste 1 com o experimento mostrado na Seção 5.1.2 mostra um ganho de 18.7%.

Tabela 5.6: Resultados do agente Avaliador executando análise independentemente do Coordenador

	α-Team (nro. agentes mineradores)	**β-Team (nro. agentes mineradores)**	**Tempo de Execução**	**Melhoria (%)**
Teste Inicial (Seção 5.1.2)	-	-	01:15:00	-
Teste 1	-	1	01:01:07	18.7
Teste 2	1	1	00:42:19	44.0

Note que, assim como mostrado na Tabela 5.6, as análises de desempenho de todos os experimentos realizados neste trabalho não consideraram as questões relativas ao sistema operacional, à quantidade de processos que estavam ativos no momento de execução dos experimentos (sistemas *multitasks*), a comunicação entre agentes, entre outros. Sendo assim, os percentuais de melhoria entre os tempos dos experimentos são até certo ponto, ilustrativos, visto não considerar todos esses aspectos mencionados. Assim, estas avaliações de tempo de execução têm a intenção apenas de demonstrar o potencial da solução no ambiente distribuído, e a vantagem da utilização das equipes de agentes, não aprofundando nos tópicos relacionados a medidas de desempenho de sistemas distribuídos.

No Teste 2 é inserido mais uma equipe de Regras de Associação, com apenas um agente de mineração. Na Tabela 5.6 pode-se notar um ganho de 44.0% no tempo em relação ao Teste 1. Quando comparamos com o Experimento mostrado na Seção 5.1.2, o ganho no tempo chega a 77.0% mostrando o potencial da abordagem de AGMI com relação à distribuição de tarefas e à execução em paralelo das mesmas.

A Seção 5.2.2 apresenta os experimentos utilizando a autonomia do agente Avaliador para melhoria da qualidade das regras.

5.2.2 Experimentos com Autonomia no Agente Avaliador

Para melhorar a qualidade das regras encontradas, foi introduzido um mecanismo de autonomia no agente Avaliador. Este mecanismo é capaz de analisar as regras produzidas, procurando o local onde há o maior número de vitórias em licitações das empresas do grupo apontado pela regra.

Nos experimentos, estabelecemos um valor constante ($c = 4$) como o número mínimo de vitórias em licitações que o grupo apontado pela regra deve ter para que a heurística seja aplicada. Desta forma, se existe algum elemento de *cluster* que agrega um número de vitórias maior ou igual à constante c, o agente calcula a frequência local da regra, e

então, após recalcular os *Rule Quality* (RQ) (Equação 4.1), o agente compara se a regra aplicada localmente teve um qualidade maior que a original ou não, registrando a melhor regra.

Vamos tomar como exemplo uma regra descoberta em um *dataset* gerado a partir de um *cluster* composto por 3 estados {MG, RJ e SP}. Se no estado de SP estiver concentrado o maior número de vitórias em licitações do grupo apontado pela regra, o agente Avaliador irá aplicar aquela regra no estado de SP e verificar se o valor de RQ aumenta, considerando a frequência local (suporte local) da regra.

Para evitar muitos acessos na base de dados, nós configuramos o agente para analisar apenas regras que tenham suporte local maior ou igual ao definido no sistema (≥ 9), obedecendo às mesmas configurações de suporte e confiança estabelecidas para as regras originais.

Experimentos com Diferentes Equipes de MD

Para validar o uso de AGMI utilizando a heurística de melhoramento de regras do agente Avaliador e as equipes de MD com diferentes números de agentes mineradores, executamos três experimentos alterando a composição das equipes de MD de Regras de Associação, considerando que este é o serviço com maior demanda no contexto do problema de cartéis em licitações. Note que na nossa arquitetura, cada equipe de MD é composta pelo Supervisor de equipe e um número de agentes de mineração, no mínimo 1.

A Tabela 5.7 mostra a composição das equipes de Regras de Associação nos experimentos executados, e o tempo gasto na execução do processo. Os experimentos foram executados utilizando também uma equipe de Clusterização, o agente Coordenador e o agente Avaliador, seguindo a mesma configuração mostrada na Tabela Tabela 5.5. A ordenação dos dados para obtenção do modelo de *clusters* foi a mesma usada no experimento da Seção 5.1.2, que gerou o modelo de *clusters* apresentado na Figura 5.1.

Tabela 5.7: Testes variando o número de agentes de Mineração nas equipes de Regras de Associação

	α-Team (nro agentes) (de Mineração.)	**β-Team (nro agentes) (de Mineração.)**	**Tempo de Execução**
Teste 1	1	1	01:01:23
Teste 2	2	1	00:42:53
Teste 3	2	2	00:42:19

Como se pode observar na Tabela 5.7, houve um ganho significativo no tempo de execução (na ordem de 30%), configurando-se dois agentes de mineração na Equipe α. No entanto, quando aumentamos a Equipe β para dois agentes, não houve praticamente nenhum ganho, apenas de alguns segundos. A explicação para esse fato leva em consideração duas tarefas executadas pelos agentes que tomaram grande parte do tempo de execução: a busca de regras de associação sob a base de dados completa e a avaliação das regras.

Considerando o experimento onde se executa apenas a tarefa de regras de associação sob a base de dados completa (Tabela 5.2), temos um tempo considerável de aproximadamente 30 minutos. Como esta é a primeira tarefa negociada pelo Coordenador, o agente que a executa fica quase todo o tempo dedicado a esta única tarefa. Quando este agente acaba sua execução, normalmente os outros agentes já terminaram as tarefas de *cluster* e as outras tarefas de regras de associação relativas aos *datasets* gerados pelos *clusters*.

O outro gargalo está no acesso ao banco de dados feito pelo agente de Avaliação. Este agente só pode iniciar suas tarefas quando as regras começam a ser gravadas na base de conhecimento. Até então, este agente deve aguardar os resultados iniciais dos outros agentes de MD para começar suas análises, terminando sempre por último. Assim, pode-se entender porque a adição de um novo agente na Equipe-β de MD não mostrou ganhos significativos em relação ao tempo de processamento. No entanto, esta é uma análise relativa à base de dados em que testamos AGMI. Neste experimento, considerando a base de dados, o modelo de *clusters* encontrado e os recursos de máquina disponíveis, três agentes de Regras de Associação foram suficientes para atingir o melhor desempenho. Com este número, temos a estabilização da curva de melhoria de tempo (Figura 5.2). Outras bases de dados que gerem diferentes configurações de *clusters* podem necessitar de mais agentes nas equipes para atingir a estabilidade na curva de tempo de processamento.

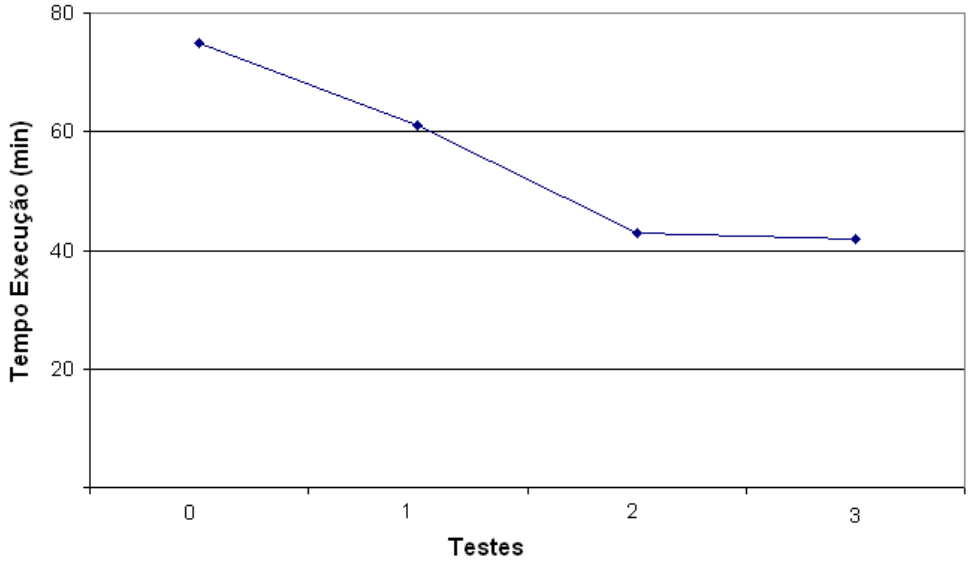


Figura 5.2: Evolução do tempo de execução utilizando agentes.

A evolução no tempo de execução do processo, aplicando a abordagem de integração de agentes com AGMI, pode ser observada na Figura 5.2. Note que o Teste 0 mostrado no gráfico é relativo ao teste apresentado na Seção 5.1.2, que usa apenas uma simples automatização de tarefas de MD de forma distribuída, sem negociação de serviços, nem heurística para melhoramento de regras. No Teste 1, usamos AGMI explorando a execução paralela do agente Avaliador (01:01:23, Tabela 5.7). Nos Testes 2 e 3, mostrados no gráfico, foram utilizados 3 e 4 agentes de Regras de Associação respectivamente para acelerar o processamento dos *datasets* gerados a partir dos *clusters* encontrados pela equipe de Clusterização (00:42:53 e 00:42:19, Tabela 5.7).

Tabela 5.8: Mecanismo de melhoramento de regras do Agente Avaliador

Informações	**Valores**
Regras Selecionadas	341
Regras Melhoradas	191
Taxa média de melhoramento	11.5%
Taxa média de melhoramento (top 10)	56.1%

Considerando o mecanismo autônomo de melhoria de regras do agente Avaliador, a Tabela 5.8 apresenta 341 regras selecionadas pelo agente para aplicação da heurística. Dentre as regras selecionadas, o agente conseguiu melhorar 191 regras. As 10 regras que mais melhoraram tiveram um aumento de 56.1% no valor de seu RQ, provando que a autonomia do agente Avaliador possibilitou a melhoria das regras.

Com isso, podemos concluir que, considerando tanto o desempenho no processamento, quanto a qualidade das regras encontradas, a abordagem de AGMI mostrou melhor resultado no nosso contexto de aplicação.

Experimento com o Primeiro Modelo de *Clusters*

No início da Seção 5.1, relatamos a alteração da ordenação dos dados a fim de obter um modelo de *clusters* diferente do encontrado no experimento apresentado na Seção 4.1.2. Desta forma, na execução dos nossos experimentos, chegamos a dois modelos de *clusters* distintos e oriundos da mesma base de dados. No entanto, os experimentos com agentes relatados até aqui fizeram uso da mesma ordenação de dados no intuito de obter o segundo modelo de *clusters* mostrado na Figura 5.1.

Neste experimento, nós utilizamos a ordenação dos dados que gerou o primeiro modelo de *clusters* (Figura 4.1) obtido no experimento de MD da Seção 4.1.2. A distribuição dos agentes é a mesma apresentada na Tabela 5.5. As Equipes α e β de regras de associação foram formadas com dois agentes de mineração cada uma. Os resultados deste experimento são apresentados na Tabela 5.9.

É importante ressaltar que neste experimento, pela primeira vez um dos agente de regras de associação utilizou sua autonomia para ajustar o valor de suporte mínimo na execução de sua tarefa. Todas as tarefas de regras de associação, com exceção da tarefa que envolve todos os dados da base, tem como suporte mínimo o valor relativo a nove ocorrências da regra. Na execução de regras de associação sob a região apontada pelo *Cluster* 1, formada pelos estados de SP, MT, MS, AL, CE, PB, PE, PI, e RN (Figura 4.1), o agente de regras de associação reajustou o suporte mínimo para $2,12\%$ do tamanho do *dataset* (849 instâncias). Esse valor de suporte mínimo reajustado permite que sejam encontradas regras com pelo menos 18 ocorrências no *dataset*.

Analisando os resultados apresentados na Tabela 5.9, pode-se notar um aumento no tempo de execução deste experimento comparado com o Experimento com diferentes equipes de MD (Teste 3 - Tabela 5.7). Note que ambos os testes são similares e utilizam as mesmas equipes de regras de associação. No entanto, o modelo de *clusters* utilizado nos testes anteriores com agentes (segundo modelo - Figura 5.1), apresentava sete regiões, enquanto que o modelo de *clusters* obtido neste experimento (primeiro modelo - Figura 4.1) é composto por 10 regiões. O acréscimo de três novas regiões no processamento

Tabela 5.9: Resultado do Experimento com o primeiro modelo de *clusters*

Informações	**Valores**
Tempo de Execução	00:52:35
Regras Encontradas	4880
Média de RQ (100 melhores regras)	57.77
Média de Suporte (100 melhores regras)	9.84
Média de RQ (10 melhores regras)	75.80
Média de Suporte (10 melhores regras)	9.60
Heurística de Melhoramento de Regras	
Informações	**Valores**
Regras Selecionadas pelo Avaliador	325
Regras Melhoradas	185
Taxa média de melhoramento	11.0%
Taxa média de melhoramento (top 10)	54.6%

justifica o aumento no tempo de processamento deste experimento. Porém, os resultados apresentados na Tabela 5.9 mostram que a qualidade das regras diminuíram em relação aos experimentos realizados, considerando o segundo modelo de *clusters* (Tabelas 5.4 e 5.8).

Experimento Consolidando Dois Modelos de *Clusters*

Neste último experimento, foi incluída uma nova equipe de clusterização a fim de se trabalhar com os dois modelos de *clusters* encontrados. Para testar o desempenho de AGMI utilizando mais de uma equipe de clusterização e comparar os resultados, adaptamos o agente Coordenador para preparar três tarefas básicas: uma de regra de associação sob a base de dados completa, e duas outras tarefas de clusterização considerando as duas ordenações distintas da base de dados já apresentadas previamente, com os respectivos modelos ilustrados na Figura 5.3. O fato de já conhecermos os modelos de *clusters* não significa que as tarefas não sejam executadas novamente. Cada execução de AGMI abrange desde o processo de preparação dos *datasets* até a avaliação das regras descobertas.

A Tabela 5.10 mostra a distribuição dos agentes entre os *hosts*, e os resultados são apresentados na Tabela 5.11. Neste experimento cada equipe de mineração foi composta pelo supervisor da equipe e um agente de MD.

A inclusão de uma nova equipe de clusterização para realização desta atividade, levando em conta distintas ordenações da base de dados, possibilitou um aumento na qualidade das regras, considerando as 100 melhores regras. Note que nesse experimento, dois modelos de *clusters* foram encontrados. O primeiro com 10 *clusters* e o segundo com sete *clusters*. Como uma das regiões encontradas em ambos os modelos eram similares, AGMI tomou apenas as 16 regiões distintas entre si, fazendo a união dos modelos encontrados. Nestas 16 regiões foram executadas as atividades de regras de associação, além da atividade de regras de associação levando em conta a base completa. Durante o processo de execução de MD nas regiões encontradas pelos *clusters*, o agente de Regras de Associação utilizou o mecanismo de reajuste de suporte duas vezes: nos processamentos dos *datasets*

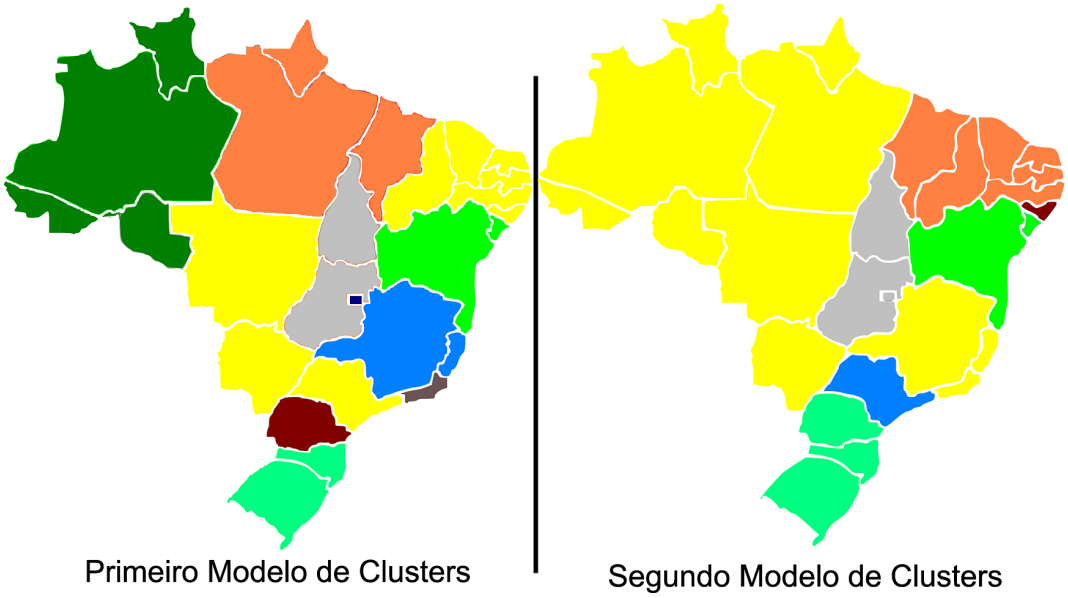


Figura 5.3: Os dois modelos de *clusters* utilizados obtidos através de diferentes ordenações

Tabela 5.10: Distribuição dos agentes - Experimento com duas equipes de Clusterização

	Host A	**Host B**
Agente Coordenador	X	
Agente Avaliador	X	
Equipe de Regras de Associação - α-Team	X	
Equipe de Regras de Associação - β-Team		X
Equipe de Regras de Associação - γ-Team		X
Equipe de Clusterização 1	X	
Equipe de Clusterização 2		X

de regras de associação oriundos do *cluster* formado pelos estados de SP, MT, MS, AL, CE, PB, PE, PI, e RN, e do *cluster* formado pelo estado de AL.

A Tabela 5.12 apresenta a evolução dos resultados dos experimentos realizados durante a realização deste trabalho de pesquisa. Pode-se notar que o experimento executado com AGMI a partir do primeiro modelo de *clusters* apresentou 4880 regras. Já os experimentos executados a partir do segundo modelo de *clusters* apresentaram 6150 regras. Note que no último experimento, integrando os dois modelos de *clusters*, 8827 regras foram encontradas. Isso mostra que muitas regras encontradas neste último experimento, integrando os dois modelos de *clusters*, foram descartadas pelo agente Avaliador como regras repetidas. Ou seja, se os modelos de *clusters* apresentassem regiões completamente distintas, o número de regras esperado seria a soma das regras obtidas na execução de MD sob os dois modelos de *clusters* encontrados, ou seja, 11030 regras. No entanto, muitas regiões formadas por ambos os *clusters* apresentam similaridades entre si, trazendo regras iguais no processo de MD.

Pode-se notar também na Tabela 5.12 que não houve aumento na qualidade das 10 melhores regras (89,70), quando comparamos este experimento integrando os dois mode-

Tabela 5.11: Resultado do Experimento com 2 equipes de clusterização

Informações	**Valores**
Tempo de Execução	01:09:28
Regras Encontradas	8827
Média de RQ (100 melhores regras)	71,90
Média de Suporte (100 melhores regras)	9,51
Média de RQ (10 melhores regras)	89,70
Média de Suporte (10 melhores regras)	9,40

los de *clusters*, com os experimentos utilizando o segundo modelo. Concluimos que as 10 melhores regras descobertas foram obtidas em uma das regiões apontadas pelo segundo modelo de *clusters*. No entanto, comparando as 100 melhores regras, o experimento integrando os dois modelos apresentou uma melhora de $3,14\%$ (71,90 - 69,71) em relação ao experimento que utilizou o segundo modelo de *clusters*, e uma melhora de $24,46\%$ (71,90 - 57,77) em relação ao experimento que utilizou a primeira ordenação para obtenção do primeiro modelo de *clusters*. Desta forma, houve uma melhora de $3,14\%$ a $24,46\%$ utilizando-se a integração dos dois modelos de *clusters* em relação aos modelos desassociados, comparando as 100 melhores regras.

Tabela 5.12: Evolução dos resultados nos experimentos

	Agente Regras de Assoc.	**AGMI 1º Modelo clusters**	**AGMI 2º Modelo clusters**	**AGMI Integração 2 Modelos**
Regras Obtidas				
Número	128	4880	6150	8827
$\overline{RQ}$ (top 100)	16.56	57.77	69.71	71.90
$\overline{RQ}$ (top 10)	33.00	75.80	89.70	89.70
Heurística de Melhoramento de Regras				
Selecionadas	-	325	341	348
Melhoradas	-	185	191	193
Melhoria média	-	11.00%	11.50%	11.72%
Melhoria (top 10)	-	54.60%	56.10%	58.48%

Houve também uma melhoria nos resultados da aplicação da heurística do agente Avaliador (Tabela 5.12). Com as regras obtidas através da integração dos modelos de *clusters*, a média de melhoria, considerando as 10 regras que o agente Avaliador mais melhorou, aumentou em $4,24\%$ ($58,48\%$ - $56,10\%$) , comparando com os experimentos realizados com segundo modelo de *clusters* e $7,11\%$ ($58,48\%$ - $54,60\%$)considerando os experimentos realizados com o primeiro modelo de *clusters*. Pode-se notar que das 348 regras selecionadas pelo agente para aplicação da heurística nos modelos integrados, 193 tiveram seu RQ aumentado, superando o resultado de todos os outros experimentos realizados.

Analisando a evolução na qualidade das regras obtidas ao longo dos experimentos, da primeira abordagem testada, utilizando apenas regras de associação, até este último

experimento, integrando modelos distintos de *clusters* com regras de associação, podemos notar um aumento de mais de 170% (89,70 - 33,00) na qualidade das 10 melhores regras, e de cerca de 335% (71,90 - 16,56) na qualidade média das 100 melhores regras. Este resultado confirma o potencial da abordagem de AGMI, que integra diferentes técnicas e modelos de MD através da atuação de agentes de software autônomos.

Na próxima seção, apresentaremos algumas regras descobertas através do processo de DCBD no AGMI, considerando a análise de especialistas da CGU.

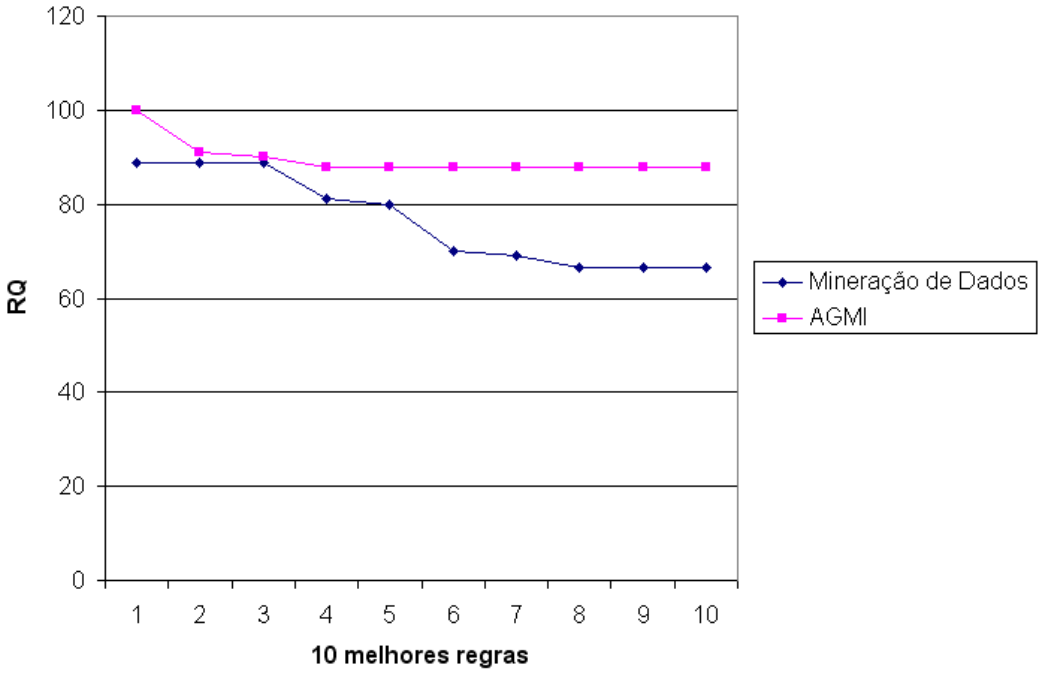


Figura 5.4: Comparação de *RQ* entre as 10 melhores regras de AGMI e DM.

5.3 Conhecimento Descoberto

Através dos experimentos com AGMI, muitas regras foram encontradas. Algumas delas foram melhoradas através da heurística do agente Avaliador. Um exemplo de regra melhorada pelo agente Avaliador é citada a seguir:

> Uma regra apontou duas empresas com 133 processos de licitações em comum. Essa grande quantidade de participações em conjunto fez dessa regra a segunda maior em suporte. No entanto, o *RQ* foi relativamente baixo, dado o número de vitórias do grupo considerando todo o território nacional. Quando o agente Avaliador aplicou a heurística nessa regra considerando o estado de Mato Grosso, o valor de *RQ* aumentou $2,71$ vezes. Isto porque todas as vitórias alcançadas pelo grupo foram no estado de Mato Grosso. O fato curioso é que nenhuma das duas empresas era de Mato Grosso, e sim de Goiás.

A regra mencionada acima mostra a importância da realização de clusterização no espaço geográfico, uma vez que as empresas que exploram o mercado de licitações não se

limitam aos seus estados de origem. Duas empresas de Goiás tiveram melhor desempenho no estado de Mato-Grosso. Outro fato é a importância da heurística do agente Avaliador testando algumas regras localmente. Possivelmente a regra citada seria descartada, considerando seu baixo valor de RQ. No entanto, com a sua aplicação local, a regra pôde melhorar no *ranking* para ser selecionada e avaliada melhor pelos especialistas.

Uma regra encontrada no experimento utilizando a ferramenta Weka, apresentado na Seção 4.1.2, foi também encontrada nos experimentos de AGMI e apareceu entre as 10 melhores regras. Segundo o especialista, essa regra aponta fortes características de cartelização e simulação de concorrência, que passaremos a descrever.

> No ano de 2008, uma empresa ganhou 9 licitações num mesmo órgão, concorrendo com outra empresa que não ganhou nenhuma das licitações em que ambas participaram. O fato curioso é que as 9 licitações que a segunda empresa perdeu foram exatamente as únicas licitações em toda a base de dados que constavam sua participação. O total de vitórias da primeira empresa na base de dados era de apenas 12, mostrando que não se tratava de um grande fornecedor.

Com relação aos resultados dos modelos de *clusters* encontrados, é interessante notar que a técnica foi inicialmente inserida no processo para possibilitar a divisão do espaço de soluções no intuito de melhorar a qualidade das regras, levando em conta que uma análise regional dos mercados de licitações poderia trazer melhores regras com maiores características de cartéis. Entretanto, os modelos de *clusters* puderam também revelar as tendências das empresas nos mercados de licitação no Brasil. Embora o algoritmo de clusterização utilizado seja sensível à ordenação dos dados, é notável o fato de que a maioria das regiões definidas pelos *clusters* encontrados em ambos os modelos era formada por estados com fronteiras em comum. Isto confirmou o fato de que apesar das empresas não se limitarem aos seus estados, normalmente elas atuam em estados próximos aos seus.

A seguir apresentaremos alguns exemplos de regras apontadas pelo AGMI para ilustrar o conhecimento descoberto no processo.

- Entre os anos de 2005 e 2007, um grupo formado por 3 empresas participou de 12 licitações. Onze dessas licitações foram organizadas por um único órgão no estado do Rio Grande do Norte nos anos de 2006 a 2007. Uma única empresa conseguiu ganhar todas estas 11 licitações. Além dessas 11 licitações, esta empresa vencedora havia ganhado somente 3 licitações na base de dados. Quando nós analisamos o histórico de participação em licitações das outras empresas do grupo, foi constatado que um dos concorrentes havia ganhado apenas 3 licitações. O outro concorrente havia ganhado 8 outras licitações, no entanto, apresentava um número razoável de participações em licitações: 72, mais que o dobro de licitações que os outros dois concorrentes já tinham participado. Note que nesta regra podem ser verificadas duas possíveis irregularidades em licitações. Há indícios de simulação de concorrência e cartelização e também de um possível direcionamento dos editais de licitação para a empresa vencedora, já que todas as vitórias foram em um órgão específico.

- Outro grupo formado por 4 empresas foi apontado por uma regra que indicou nove processos de licitação em comum. Esta regra foi encontrada na região formada pelos estados de Paraná e Rio Grande do Sul. Neste grupo, duas empresas destacaram por ganhar 8 licitações. Uma das empresas ganhou 6 e a outra ganhou 2 licitações. Na

análise do histórico de vitórias em licitações, foi verificado que a primeira empresa tinha ganhado apenas outras 3 licitações no banco de dados inteiro, e a segunda empresa tinha ganhado apenas 2 outras licitações. As outras duas empresas do grupo apontado pela regra eram empresas de grande porte, e apresentavam um histórico considerável de participações em licitações. Desta forma, as duas empresas entraram na regra pelo simples fato de participarem com frequência de licitações, não evidenciando característica de cartel. No entanto, as participações das duas primeiras empresas já configuraram um cenário de alerta em relação a cartel.

- Outra regra apontou duas empresas que participaram juntas de 10 licitações entre os anos de 2007 a 2009. As licitações em comum ocorreram nos estados de Pernambuco e Rio Grande do Norte. Dessas licitações em comum, uma das empresas ganhou 6 e a outra não ganhou nenhuma. As 6 licitações que a primeira empresa ganhou foi em um único órgão no estado do Rio Grande do Norte. A análise do histórico de participações mostrou que a empresa vencedora só havia participado dessas exatas 10 licitações, enquanto que a segunda já havia participado de outros 15 certames, vencendo um. Neste caso, chama à atenção a quantidade atípica de vitórias da primeira empresa. No caso dessa regra, os indícios apontam para direcionamento de edital e simulação de concorrência.

- Outras três empresas foram apontadas por uma regra com 14 licitações em comum. Estas empresas ganharam juntas 8 dessas 14 licitações. Na análise do histórico de participações em licitações nós encontramos uma média de 30 licitações por empresa. Uma média relativamente baixa de participações em licitações, considerando o número de vitórias nas licitações em comum.

- Duas empresas de um mesmo estado, sendo que o total de licitações que cada uma participou individualmente foi de 75 e 78. Dessas licitações, as empresas participaram juntas de 68 ganhando 14 contratos entre os anos de 2005 a 2007.

Na análise das regras, é sempre considerada a quantidade de vitórias que uma empresa consegue em licitações, uma vez que a média de vitórias das empresas é relativamente baixa. Quando consideramos o contexto de nossa base de dados, na média uma empresa vence $16,8\%$ das licitações de que participa. No caso das empresas que já ganharam pelo menos uma licitação (1162 empresas), a média de vitórias é de $2,24$ licitações. Outro fato que deve ser considerado na análise de uma empresa apontada por uma regra é a sua frequência de participação em licitações. Vamos tomar como exemplo o caso da empresa que mais ganhou licitações em nossa base de dados. Essa empresa ganhou um total de 42 licitações. No entanto, no seu histórico de participação de licitações na base de dados, constam 358 diferentes licitações, mantendo uma proporção de vitórias abaixo da média ($\approx 12\%$). Neste caso, trata-se possivelmente de um grande fornecedor, que atua com frequência no ramo de licitações. No entanto, quando uma empresa consegue 7 ou 9 contratos por exemplo, especialmente quando os contratos foram em um único órgão, levando em conta ainda que sua participação nas licitações da base de dados não é tão frequente, o comportamento atípico apresentado pode ser indício de uma irregularidade e deve ser averiguado.

Quando consideramos a frequência de participações de uma empresa em licitações, a média de participação de uma empresa em nossa base de dados é 21, levando em conta

as empresas que participaram de 5 ou mais licitações (1085 empresas). Se consideramos então somente as empresas que participaram de pelo menos 9 licitações, número este definido como suporte mínimo nos nossos testes, essa média sobe para aproximadamente 30 licitações. Note que quando comparamos a média de vitórias e média de participações em licitações, os números apresentados pelas empresas apontadas nas regras de AGMI mostram realmente um desequilíbrio nessa proporção, evidenciando a provável presença de irregularidades no processo. Além da atipicidade de atuação das empresas suspeitas nas licitações, outros aspectos são considerados na análise preliminar, como apresentado na Seção 3.1.3. Desta forma, pela avaliação preliminar das regras, e os indícios encontrados, o conhecimento descoberto mostrou que a técnica apresentada realmente traz bons resultados com relação à procura de grupos suspeitos de prática de cartel.

Em suma, as análises preliminares realizadas com ajuda do especialista, consideram principalmente a atipicidade do desempenho das empresas suspeitas nas licitações em que participam e também a taxa de sucesso dessas empresas nas licitações. Outros aspectos considerados na análise são os apresentados em Justiça (2008), tais como:

- Desistência de fornecedores inesperadamente do processo de licitação;
- Um determinado concorrente vencendo muitas licitações que possuem a mesma característica ou que se referem a um tipo especial de contratação.
- Existência de um concorrente que sempre oferece propostas, apesar de nunca vencer as licitações.

Além disso, a experiência do especialista no tratamento do problema é utilizada nas análises dos grupos suspeitos.

Outras regras foram apontadas por AGMI revelando grupos de empresas com reais características de cartel. Entretanto, algumas outras regras encontradas por AGMI e classificada como boas regras não apresentaram características de cartel. Em nossa análise, isso decorreu da função de avaliação adotada. Apesar da função considerar frequência de participação do grupo e número de vitórias em licitações, outros parâmetros de análise devem ainda ser incorporados para melhorar a avaliação. E para isso, a participação de especialistas no processo é fundamental.

Uma outra dificuldade encontrada no processo de DCBD é que muitas regras de boa qualidade apresentadas no relatório final são na verdade derivadas de outras. Suponha uma regra formada pelas empresas A, B e C com características de cartel. Caso as licitações comuns dessas empresas tenham sido frequentadas por grandes fornecedores, por exemplo as empresas D e E, outras regras consideradas de igual qualidade poderão ser relatadas, tal como um grupo formado por A, B, C, D e E, ou o grupo formado por A e B. Note que alguns subconjuntos e superconjuntos de A, B e C poderão figurar como boas regras. De fato elas tratarão de praticamente os mesmos processos de licitações, no entanto, a avaliação das regras precisa ser trabalhada para que esse tipo de filtragem já seja realizado pelo agente Avaliador, para facilitar a análise do especialista.

É interessante notar que as regras relatadas como exemplo, foram extraídas do relatório de AGMI, que as apresentou dentre as melhores regras, num procedimento totalmente automatizado. As regras foram encaminhadas para a Secretaria de Prevenção à Corrupção para uma averiguação mais apurada.

Capítulo 6

Conclusão e Trabalhos Futuros

Além dos trabalhos de auditoria e fiscalização, no âmbito do Poder Executivo Federal, a CGU é também responsável por desenvolver mecanismos de prevenção a corrupção. Desta forma, além de detectar e punir os casos de corrupção, a CGU deve desenvolver mecanismos capazes de evitar ou dificultar a ocorrência de corrupção ou mesmos irregularidades no funcionamento do Administração Pública. Nesse contexto, a formação de cartéis em licitações públicas é um problema grave de corrupção, a qual deve ser considerada. Assim, este trabalho apresentou uma proposta de integração das áreas de MD e SMA e sua aplicação no domínio de auditoria governamental. Nesta proposta, foi apresentado AGMI, uma arquitetura integrando agentes de mineração, e sua implementação utilizando a plataforma JADE, com o desenvolvimento do *DMA Framework*, uma extensão do *framework* Weka.

AGMI é uma arquitetura de três camadas (estratégica, tática e operacional) com quatro diferentes tipos de agentes (Coordenador, Avaliador, Supervisor de Equipes de MD e Agentes de MD). AGMI foi definida para aplicar diferentes técnicas de MD utilizando uma abordagem cooperativa de integração entre os agentes, para operar num ambiente distribuído, com uma perspectiva integrada, onde o objetivo principal é melhorar o conhecimento descoberto.

Além disso, a AGMI provê a automatização do processo de preparação dos dados, utilizando a abordagem de agentes inteligentes para construção dos *datasets* iniciais, para execução das primeiras tarefas definidas no processo, e para construção de *datasets* dinâmicos, baseados nos resultados obtidos por agentes de MD. Além disso, os agentes de MD utilizam autonomias para reajustar parâmetros de configuração adaptando-se aos recursos disponíveis para processamentos de seus algoritmos, tornando a ferramenta flexível para operar em bases de dados de tamanhos variados independente do recurso disponível.

Para desenvolvimento de AGMI foi feito um estudo de integração das áreas de MD e SMA, seguindo a linha de pesquisa de AMII. Foram estudados algoritmos de MD disponíveis na ferramenta Weka, para o desenvolvimento do *DMA Framework*, o qual pode ser utilizado independente de AGMI para execução de algoritmos de MD e manipulação das estruturas e modelos encontrados como resultados para outras aplicações de SMA. A plataforma e o *framework* JADE foram também estudados a fim de possibilitar o desenvolvimento do protótipo utilizando a abordagem de SMA.

Neste trabalho, aplicamos AGMI como uma proposta de solução para o problema de detecção de cartéis em licitações públicas em dados reais extraídos do Sistema Compras-

Net. Muitos experimentos foram feitos e os resultados apresentados na Seção 5 mostraram o potencial da solução proposta para tratamento deste problema. Os resultados experimentais apresentados nas Figuras 5.2 and 5.4 mostraram também que AGMI teve um bom desempenho em termos de performance e produção de regras de qualidade. A heurística utilizada autonomamente pelo agente de Avaliação também mostrou ser eficaz melhorando, numa média de $11,72\%$, a qualidade de 193 regras, dentre as quais, as dez melhores tiveram uma média de melhoria de $58,48\%$, utilizando a integração de dois modelos de clusters (Tabela 5.12).

Várias Regras de Associação foram descobertas trazendo indícios de irregularidades em licitações. Nas análises preliminares dos especialistas, as regras apresentadas por AGMI trouxeram grupos de empresas atuando de forma atípica e com indícios de práticas de cartéis, simulação de concorrência em licitações e até mesmo direcionamento de editais de licitações. Em suma, o conhecimento descoberto, apresentado na Seção 5.3, pode auxiliar o trabalho dos auditores do Governo Federal a detectar e também a prevenir a ocorrência não só de cartéis como também de outras irregularidades durante os processos futuros de licitação.

Ressaltamos que o problema de formação de cartéis é muito importante ser tratado, pois reflete não somente um grande prejuízo ao bom andamento das atividades da Administração Pública, mas também à vida dos empresários e dos cidadão comuns, cujos impostos são aplicados e, muitas vezes, desperdiçados nos superfaturamentos de obras prejudicadas por licitações eivadas de vícios.

Em trabalhos futuros, além da descoberta de regras, AGMI também poderá ser trabalhado para integrar futuramente o ComprasNet com a finalidade de averiguar os processos de licitação em tempo real, expedindo alertas ao responsável pela licitação e à controladoria do órgão, evitando assim as possíveis irregularidades no processo. Da mesma forma, serão avaliadas as possibilidades de aplicação de AGMI em outros sistemas do governo federal para geração de conhecimento de auditoria e posterior monitoramento dos gastos públicos com finalidade preventiva.

Ademais, no âmbito da CGU, além do uso de AGMI nas diversas bases de licitação para subsidiar as auditorias, os resultados dos processamentos poderão ser usados para identificar regiões de risco, onde ocorrem maiores indícios de irregularidades. Isto é extremamente útil na fase de prevenção à corrupção e possibilita que os órgãos de controle desenvolvam mecanismos específicos nessas regiões, evitando assim os possíveis prejuízos advindos de ações fraudulentas nos processos licitatórios.

Posteriormente serão avaliadas formas de automatização da seleção de atributos para realização de divisão do espaço de soluções, além da divisão geográfica feita por clusterização, no intuito de melhorar ainda mais o conhecimento descoberto. Além disso, serão estudadas, juntamente com os especialistas em auditoria pública, novas formas de avaliar automaticamente os modelos de cartéis descobertos, com o objetivo de facilitar a análise final do conhecimento descoberto. Neste sentido, outras autonomias serão introduzidas nos agentes de AGMI, almejando o enriquecimento das regras.

Além disso, o AGMI pode ser aplicado em diferentes domínios da área de auditoria governamental para descoberta de conhecimentos úteis, para prevenção e detecção de irregularidades e para fraudes no âmbito da prevenção e do combate à corrupção. Para isso, ontologias serão utilizadas no aperfeiçoamento do tratamento semântico de várias bases de dados distribuídas, através de agentes de mineração executando diferentes técnicas de

MD. Outro aspecto a ser estudado é o aumento da autonomia dos agentes através de mecanismos de aprendizagem de máquina e de um protocolo especializado de interação de agentes que permita a otimização das atividades de descoberta de conhecimento através da aplicação do modelo definido.

6.1 Publicações

Alguns resultados desta pesquisa foram adaptados e transformados em artigos científicos. A seguir, são listados os trabalhos publicados, aceitos para publicação ou ainda em avaliação até a data presente.

- C. V. S. Silva e C. G. Ralha. Utilização de técnicas de mineração de dados como auxílio na detecção de cartéis em licitações. In XXX Congresso da Sociedade Brasileira de Computação (SBC), II Workshop de Computação Aplicada em Governo Eletrônico (WCGE), 2010. ISSN 2175-2761, Retrieved July 23, 2010 from `http://www.inf.pucminas.br/sbc2010/anais/wcge/index_arquivos/artigos_1.htm`.
- C. V. S. Silva e C. G. Ralha. AGMI: An AGent-Mining Tool and its Application to Brazilian Government Auditing. In WEBIST (1), 2011b. No prelo.
- C. V. S. Silva, C. G. Ralha, e H. A. Rocha. Técnicas de mineração de dados como apoio as auditorias governamentais. Revista CGU, 8(1), 2011. No Prelo.
- C. V. S. Silva e C. G. Ralha. An agent mining approach applied to brazilian government auditing. JAAMAS - Journal of Autonomous Agents and Multi-Agent Systems, 2011a. Em avaliação.
- C. V. S. Silva e C. G. Ralha. Detecção de Cartéis em Licitações Públicas com Agentes de Mineração de Dados. RESI - Revista Eletrônica de Sistemas de Informação. Em avaliação.

Referências

R. Agrawal e R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8. 15

K. A. Albashiri, F. Coenen, e P. H. Leng. Emads: An extendible multi-agent data miner. *Knowl.-Based Syst.*, 22(7):523–528, 2009. 44

M. T. De Araujo e A. Martinez. Para combater infrações econômicas em licitações. Jornal Valor Econômico, July 2010. 40

Bahia. Proposta de anteprojeto: Normas de auditoria governamental - nags aplicáveis ao controle externo. Technical report, Tribunal de Contas do Estado da Bahia, 2007. 34

F. L. Bellifemine, G. Caire, e D. Greenwood. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. Wiley, April 2007. x, xi, 26, 27, 28, 65, 106, 107, 108, 109

D. Bertolini, A. Novikau, A. Susi, e A. Perini. Taom4e: an eclipse ready tool for agent-oriented modeling. issue on the development process. Technical report, Fondazione Bruno Kessler - irst, 2006. 32

S. Borman. The expectation maximization algorithm a short tutorial. em-tut@seanborman.com, 2004. 14

Brasil. Constituição da república federativa do brasil de 1988. D.O.U. de 5/10/1988, 1988. 34, 36

Brasil. Lei n. 10.683, de 28 de maio de 2003. D.O.U. de 29/05/2003, 2003. Disponível em: `http://www.planalto.gov.br/CCIVIL/leis/2003/L10.683.htm`. Acesso: 16 fev. 2011. 35

Brasil. Lei n. 8.666, de 21 de junho de 1993. D.O.U. de 22/06/1993, 1993. Disponível em: `http://www.planalto.gov.br/ccivil_03/LEIS/L8666cons.htm`. Acesso: 16 fev. 2011. 36

P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, e A. Perini. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. 28, 30

R. N. Carvalho, K. B. Laskey, P. C. G. Da Costa, M. Ladeira, L. L. Santos, e S. Matsumoto. Probabilistic ontology and knowledge fusion for procurement fraud detection in brazil. In F. Bobillo, P. C. G. Da Costa, C. D'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, T. Martin, M. Nickles, M. Pool, e P. Smrz, editors, *URSW*, volume 527 of *CEUR Workshop Proceedings*, pages 3–14. CEUR-WS.org, 2009. 42

CGU. Controle interno, prevenção e combate à corrupção - ações da cgu em 2008, 2008. Disponível em: `http://www.cgu.gov.br/Publicacoes/BalancoAcoes2008/Arquivos/balanco2008_portugues.pdf`. Acesso: 16 fev. 2011. 35

P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, e R. Wirth. Crisp-dm 1.0 step-by-step data mining guide. Technical report, The CRISP-DM consortium, August 2000. Disponível em: `http://www.crisp-dm.org/CRISPWP-0800.pdf`. Acesso: 16 fev. 2011. x, 7, 8, 43

P. Cheeseman e J. Stutz. Bayesian classification (autoclass): theory and results. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, e Ramasamy Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 153–180. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. ISBN 0-262-56097-6. 10

75 contributors. *Global Corruption Report 2009-Corruption and the Private Sector*. Cambridge University Press, 2009. ISBN 9780521132404. 42

K. H. Dam e M. Winikoff. Comparing agent-oriented methodologies. In P. Giorgini, B. Henderson-Sellers, e M. Winikoff, editors, *AOIS*, volume 3030 of *Lecture Notes in Computer Science*, pages 78–93. Springer, 2003. 28, 29

A. P. Dempster, N. M. Laird, e D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. 14

C. B. Do e S. Batzoglou. What is the expectation maximization algorithm? *Nature Biotechnology*, 26(8):897–899, August 2008. ISSN 1087-0156. doi: 10.1038/nbt1406. x, 12, 13, 14

U. M. Fayyad, G. Piatetsky-Shapiro, e P. Smyth. From data mining to knowledge discovery: an overview. In *Advances in knowledge discovery and data mining*, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. x, 6, 7, 21

R. J. Ferreira. *Auditoria*. Editora Ferreira, Rio de Janeiro, RJ - Brasil, 2007. 33

FIPA. FIPA ACL message structure specification. FIPA agent communication language specifications, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, 2002a. Disponível em: `http://www.fipa.org/specs/fipa00061`. Acesso: 16 fev. 2011. 22

FIPA. FIPA Communicative Act Library Specification. FIPA agent communication language specifications, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, December 2002b. 22

FIPA. FIPA contract net interaction protocol specification. Interaction protocols, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, 2002c. Disponível em: http://www.fipa.org/specs/fipa00029/. Acesso: 16 fev. 2011. 27

FIPA. FIPA agent management specification. Component, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, 2004. Disponível em: http://www.fipa.org/specs/fipa00023/. Acesso: 16 fev. 2011. 107

FIPA. FIPA ontology service specification. FIPA agent communication language specifications, FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS, 2001. Disponível em: http://www.fipa.org/specs/fipa00086/XC00086D.html. Acesso: 16 fev. 2011. x, 24

W. J. Frawley, P. G. Shapiro, e C. J. Matheus. Knowledge discovery in databases - an overview. *Ai Magazine*, 13:57–70, 1992. 6

J. Gao, J. Denzinger, e R. C. James. A cooperative multi-agent data mining model and its application to medical data on diabetes. In V. Gorodetsky, J. Liu, e V. A. Skormin, editors, *AIS-ADM*, volume 3505 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2005. ISBN 3-540-26164-8. 43

P. Giorgini, M. Kolp, e J. Mylopoulos. Agent oriented software development. In *In Methods and Applications of Artificial Intelligence*, 2002. 27

P. Giorgini, J. Mylopoulos, e R. Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Engineering Applications of Artificial Intelligence*, 18:159–171, 2005. x, 28, 29, 30

J. Han e M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. ISBN 1558609016. 7, 12, 15

B. Henderson-Sellersa e P. Giorgini. *Agent-oriented methodologies*. Idea Group Pub., 2005. 27

J. Hendler. Is there an intelligent agent in your future? *Nature*, 1999. 23

M. N. Huhns e L. M. Stephens. *Multiagent systems and societies of agents*, pages 79–120. MIT Press, Cambridge, MA, USA, 1999. 21, 23

A. K. Jain e R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988. ISBN 0-13-022278-X. 10

A. K. Jain, M. N. Murty, e P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), Setember 1999. 11

S. Jund. *Auditoria: conceitos, normas, técnicas e procedimentos*. Elsevier, Rio de Janeiro, RJ - Brasil, 2005. 33

Ministério Justiça. Combate a cartéis em licitações. Publicação Oficial, 2008. 40, 41, 42, 94

K. Karoui, F. B. Ftima, e H. B. Ghezala. A multi-agent framework for anomalies detection on distributed firewalls using data mining techniques. In Longbing Cao, editor, *Data Mining and Multi-agent Integration*, pages 267–278. Springer US, 2009. ISBN 978-1-4419-0522-2. 42, 43

H. Lee. Agent based video contents identification and data mining using watermark based filtering. In Longbing Cao, editor, *Data Mining and Multi-agent Integration*, pages 315–324. Springer US, 2009. ISBN 978-1-4419-0522-2. 42

O. Marban, G. Mariscal, e J. Segovia. A data mining and knowledge discovery process model. In Julio Ponce e Adem Karahoca, editors, *Data Mining and Knowledge Discovery in Real Life Applications*. IN-TECH 2009, 2009. 8

M.Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, e I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009. 103

Sistema Brasileiro de Defesa da Concorrência Ministério da Justiça. Cartel. http://www.mj.gov.br, 2009. Disponível em: `http://www.mj.gov.br/sde/data/Pages/MJ9F537202ITEMIDDEB1A9D4FCE04052A5D948E2F2FA2BD5PTBRNN.htm`. Acesso: 16 fev. 2011. 40

C. Moemeng, V. Gorodetsky, Z. Zuo, Y. Yang, e C. Zhang. Agent-Based Distributed Data Mining: A Survey. In Longbing Cao, editor, *Data Mining and Multi-agent Integration*, chapter 3, pages 47–58. Springer US, Boston, MA, 2009. 21

N. F. Noy e L. D. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. Online, 2001. Disponível em: `http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html`. Acesso: 16 fev. 2011. 24

M. S. Z. Di Pietro. *Direito Administrativo*. Atlas, 22 edition, 2009. 34, 36, 37, 39

PMBOK. *A Guide to the Project Management Body of Knowledge*. Project Management Institute, 2010. 57

D. Prescher. A Tutorial on the Expectation-Maximization Algorithm Including Maximum-Likelihood Estimation and EM Training of Probabilistic Context-Free Grammars. *CoRR*, abs/cs/0412015:49, 2004. Presented at the 15th European Summer School in Logic, Language and Information (ESSLLI 2003). 14

C. G. Ralha. Towards the integration of multiagent applications and data mining. In Longbing Cao, editor, *Data Mining and Multi-agent Integration*. Springer US, 2009. 2

C. G. Ralha, H. W. Schneider, L. O. Da Fonseca, M. E. M. T. Walter, e M. M. Brigido. Using bioagents for supporting manual annotation on genome sequencing projects. In A. L. C. Bazzan, M. Craven, e N. F. Martins, editors, *BSB*, volume 5167 of *Lecture Notes in Computer Science*, pages 127–139. Springer, 2008. ISBN 978-3-540-85556-9. 42

S. J. Russell e P. Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 1st edition edition, January 1995. 20

S. J. Russell e P. Norvig. *Artificial Intelligence: A Modern Approach*. Publisher: Prentice Hall, 3nd edition edition, 2010. x, 17, 18, 20

Secretaria do Tesouro Nacional. Estatísticas de uso, 2011. Disponível em: `https://consulta.tesouro.fazenda.gov.br/estatisticas/index_estatistica_menu.asp`. Acessado em 10/02/2011. 1

C. V. S. Silva e C. G. Ralha. Utilizacao de tecnicas de mineracao de dados como auxilio na deteccao de carteis em licitacoes. In *WCGE: II Workshop de Computação Aplicada em Governo Eletronico*. Sociedade Brasileira de Computacao, 2010.

R. G. Smith. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *Transactions on Computers*, C-29(12):1104–1113, 1980. 27

P. Tan, M. Steinbach, e V. Kumar. *Introduction to Data Mining*. Addison Wesley, us ed edition, May 2005. ISBN 0321321367. 9, 10, 11, 15, 16, 21, 79

G. Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-23203-0. 17

Weka. Note on randomization, 2010. Disponível em: `http://weka.wikispaces.com/Use+WEKA+in+your+Java+code`. Acesso: 16 fev. 2011. 78, 79

I. H. Witten e E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005. ISBN 0120884070. x, 2, 11, 12, 15

I. H. Witten e E. Frank. Data mining: practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record*, 31(1):265–320, 2002. 103, 104

M. Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons, June 2002. ISBN 047149691X. 17, 20, 21, 22, 23, 25, 26

M. Wooldridge e N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10:115–152, 1995. 17

X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, e D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, 2007. ISSN 0219-1377. 11, 14

H. B. Zghal, S. Faiz, e H. B. Ghezala. A Framework for Data Mining Based Multi-Agent: An Application to Spatial Data. In *WEC'05: The Third World Enformatika Conference*, 2005. 42

Apêndice A

Tecnologias Utilizadas

A.1 Weka

Weka é uma ferramenta de código aberto e desenvolvido sob licença *General Public Licence* (GNU), na Universidade de Waikato, sobre a plataforma Java. O software reúne uma coleção de algoritmos de aprendizagem de máquina, direcionados a tarefas de MD. Os algoritmos podem ser aplicados diretamente em um conjunto de dados, ou podem ser chamados a partir de aplicações Java. O produto engloba funcionalidades de pré-processamento, classificação, regressão, clusterização e regras de associação. Além do software ter uma interface gráfica amigável, disponibiliza suas funcionalidade para serem iniciadas pelo console ou para serem acessadas via código Java a partir de outras aplicações. Isto torna a ferramenta bastante flexível e poderosa (M.Hall et al., 2009).

Atualmente, o Weka contém mais de 70 algoritmos de classificação e regressão, diversos outros de associação, agrupamento, ferramentas de pré-processamento de dados, seleção de atributos, além de visualizadores gráficos que permitem a melhor compreensão dos dados.

A Figura A.1 mostra a interface gráfica do Weka. As abas dividem as classes de funcionalidades em:

- Pré-processamento - escolher e modificar o dado que está sendo trabalhado.
- Classificação - treinamento e teste de esquema de aprendizagem para classificar dados ou efetuar regressão.
- Agrupamento - aprender agrupar dados.
- Associação - aprender regras de associação para dados.
- Seleção de Atributos - selecionar a maioria dos atributos relevantes nos dados.
- Visualização - visualizar uma plotagem bidimensional dos dados.

A partir das abas é possível selecionar o algoritmo que o usuário deseja, ajustando os valores de seus parâmetros. A ferramenta está apta a trabalhar tanto com arquivos (texto, csv) ou utilizar diretamente com um banco de dados, através da interface JDBC (Witten e Frank, 2002).

Por padrão, a aplicação vem configurada para executar com no máximo 256M de memória. No entanto, devido à complexidade da maioria dos algoritmos de mineração de

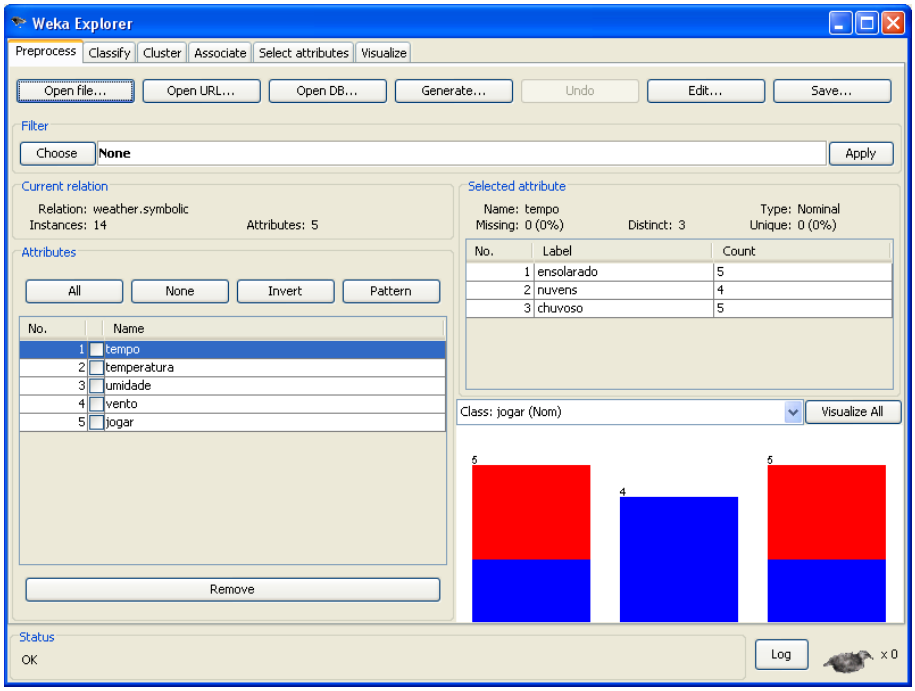


Figura A.1: Inteface Gráfica do Weka

dados, pode ser necessário a configuração do arquivo RunWeka.ini da aplicação para que esta possa ser executada com mais memória disponível.

Dataset é um dos conceitos mais básicos em aprendizagem de máquina. É nada mais que uma planilha de dados, cada linha contendo diversas colunas de atributos (Witten e Frank (2002)). No Weka, o *dataset* tem um formato específico, seguindo o padrão *Attribute-Relation File Format* (ARFF). O ARFF é um arquivo texto ASCII que descreve uma lista de instâncias, compartilhando um conjunto de atributos. Esse formato foi desenvolvido pelo *Machine Learning Project*, no Departamento de Ciência da Computação da Universidade de Waikato, para uso com Weka [1].

Pode conter atributos do tipo numérico,*string*, data, e também pode ser do tipo classe (uma lista predefinida de valores). A seguir, pode-se observar um *dataset* no formato ARFF com os valores mostrados na Tabela 2.1.

```
%nome do dataset
@relation tempo_jogar

%atributos e seus possíveis valores
@attribute tempo {ensolarado, nuvens, chuvoso}
@attribute temperatura {quente, moderado, frio}
@attribute umidade {alta, normal}
@attribute vento {verdadeiro, falso}
@attribute jogar {sim, nao}

%dados
@data
ensolarado,quente,alta,falso,nao
```

[1] Para mais informações, ver `http://www.cs.waikato.ac.nz/~ml/weka/arff.html`

```
ensolarado,quente,alta,verdadeiro,nao
nuvens,quente,alta,falso,sim
chuvoso,moderado,alta,falso,sim
chuvoso,frio,normal,falso,sim
chuvoso,frio,normal,verdadeiro,nao
nuvens,frio,normal,verdadeiro,sim
ensolarado,moderado,alta,falso,nao
ensolarado,frio,normal,falso,sim
chuvoso,moderado,normal,falso,nao
ensolarado,moderado,normal,verdadeiro,sim
nuvens,moderado,alta,verdadeiro,sim
nuvens,quente,normal,falso,sim
chuvoso,moderado,alta,verdadeiro,nao
```

A.2 JADE

JADE é um framework, implementado em Java, que simplifica a implementação de SMA através de um *middleware* que cumpre as especificações da FIPA (`http://jade.tilab.com/`). Através do JADE, os agentes podem ser distribuídos em máquinas diferentes, independente do sistema operacional. E devido a sua interface gráfica para gerenciamento dos agentes, estes podem ser configurados remotamente. JADE é também software livre e é distribuído sob os termos da licença LGPL (*Lesser General Public License Version 2*).

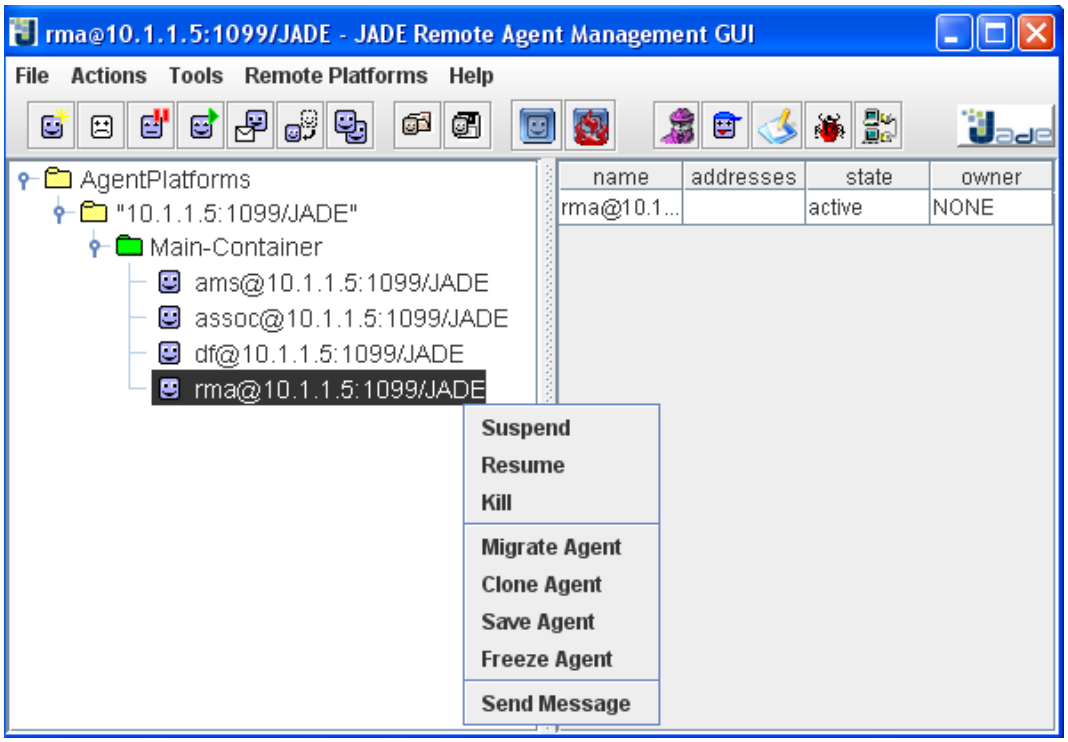


Figura A.2: Interface gráfica do JADE 4.0

A Figura A.2 apresenta a interface gráfica do JADE 4.0. É possível interagir com o agente através de envio de mensagens e alterar também seus estados. Além disso, várias

outras funcionalidades são disponibilizadas na ferramenta. Uma outra forma de trabalhar com os agentes, talvez mais usual, é inicializando-os no JADE através do console. A Figura A.3 mostra a inicialização de um agente de mineração no contêiner principal do framework.

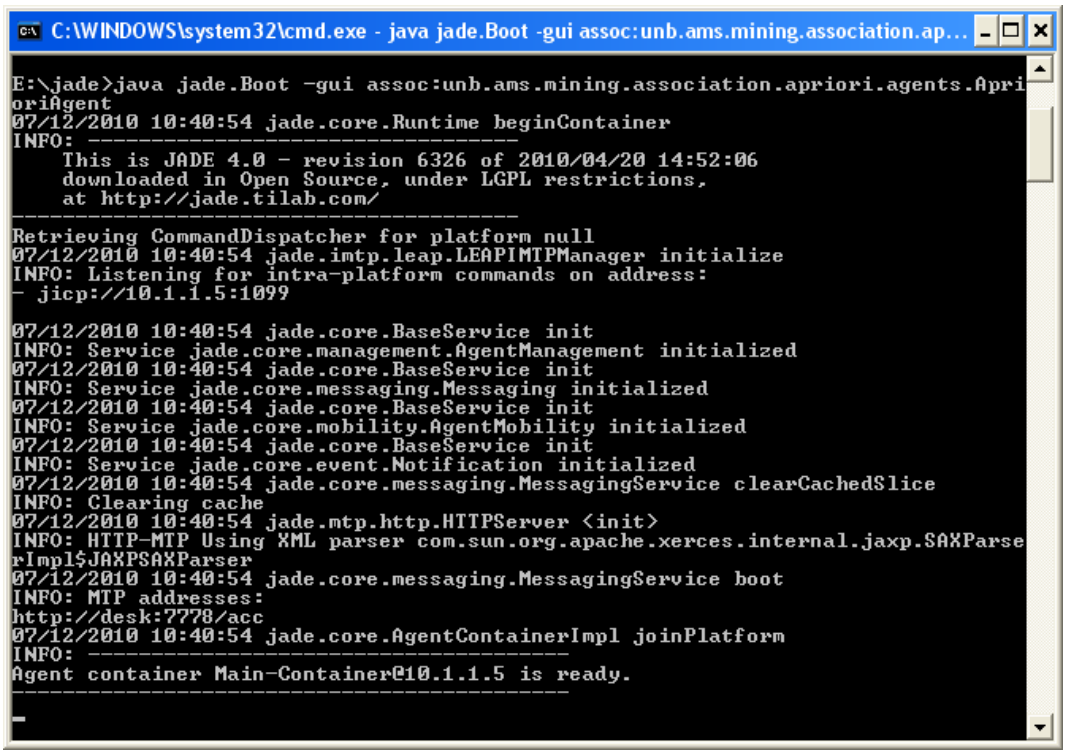


Figura A.3: Inicialialização de um agente no JADE através do console

Algumas características do JADE são citadas em Bellifemine et al. (2007), tais como: ambiente completamente distribuído, completa conformidade com as especificações FIPA, transporte eficiente de mensagens assíncronas, implementação de páginas amarelas, gerenciamento simples do ciclo de vida dos agentes, suporte a mobilidade de agentes, interface gráfica, suporte a ontologias e linguagens de conteúdo, biblioteca de protocolos de interação, entre outros. JADE é um framework bastante utilizado no desenvolvimento de SMA e já homologado pela comunidade científica.

A arquitetura do JADE é baseada num contêiner principal, chamado *Main Container*. Esse contêiner é sempre iniciado, e todos os outros contêineres se ligam a ele na medida em que são carregados. Para isso, os contêineres se registram no contêiner principal. É através desses contêineres que a plataforma JADE possibilita a distribuição dos agentes em máquinas diferentes. A Figura A.4 mostra o Contêiner-1 e Contêiner-2 carregados em hosts diferentes e ligados ao contêiner principal (*main container*).

Algumas responsabilidades do contêiner principal do JADE são citadas em Bellifemine et al. (2007). Este contêiner é responsável por gerenciar a *Containers Table* (CT), que é a tabela de contêineres, gerenciar a *Global Agent Descriptor Table* (GADT), que é a tabela com o registro de todos os agentes presentes na plataforma com seus atuais estados e localização. Também nesse contêiner são hospedados dois agentes especiais, *Agent Management System* (AMS) e DF, responsáveis pelas páginas brancas (registro

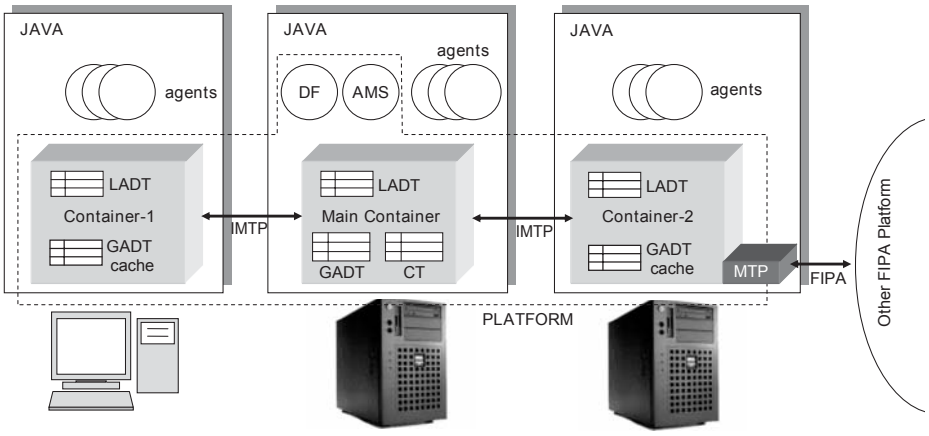


Figura A.4: Relação entre os elementos arquiteturais do JADE (Bellifemine et al., 2007).

dos agentes) e amarelas (registro dos serviços), respectivamente, na plataforma. Esses agentes podem ser vistos na Figura A.2, iniciados no contêiner principal.

O agente AMS é responsável por supervisionar toda a plataforma. Ele é o contato de todos os agentes que necessitam interagir para acessar as páginas brancas da plataforma e gerenciar seus ciclos de vida. Todo agente precisa se registrar com o AMS para obter um identificador válido. Isso é feito na inicialização de cada agente. Já o agente DF é um agente que implementa o serviço de páginas amarelas, usado por aquele agente que deseja registrar seus serviços ou procurar por outros serviços disponíveis. O agente DF também aceita registro de agentes que desejam ser notificados quando houver um registro de um serviço. Outros agentes DF podem ser iniciados para distribuir as páginas amarelas em diferentes domínios (Bellifemine et al., 2007). Descrição similar é apresentada na especificação de FIPA de gerenciamento de agentes (FIPA, 2004).

Desenvolvimento de um agente no JADE

Um agente na plataforma JADE é criado estendo-se a superclasse *Agent*. Esta classe é a superclasse comum para definição de agentes e provê métodos para realização das tarefas básicas de agentes, tais como: troca de mensagens; realização do ciclo de vida do agente, incluindo a inicialização, suspensão e finalização do agente; e também o agendamento e execução de múltiplas e concorrentes atividades, implementadas através dos comportamentos.

Agent já tem pré-definidos os métodos *setup()* e *takeDown()*, que podem ser sobrescritos para definir as ações de inicialização e finalização do agente, respectivamente. A definição do método *setup()* permite ao desenvolvedor prover o comportamento necessário do agente. Quando este método é chamado, o agente já foi registrado na plataforma e já está habilitado a enviar e receber mensagens. No entanto, o modelo de execução é ainda sequencial, não possuindo ainda o agente nenhum tipo de comportamento concorrente ativo. O método *takeDown()* é o último método a ser executado no agente. Este é chamado quando o agente está sendo destruído. Desta forma, devem ser implementadas neste método as atividades finais do agente. A seguir, um exemplo simples de implementação de um agente JADE.

```
package unb.agmi.agents;
```

```
2 import jade.core.Agent;
  public class AgentExample extends Agent{
4   @Override
    protected void setup() {
6     System.out.println("Eu sou um agente!");
    }
8   @Override
    protected void takeDown() {
10    System.out.println("Fim!");
    }
12 }
```

Este agente imprimirá as mensagens "Eu sou um agente!", e logo em seguida "Fim!". Para se construir um agente capaz de realizar tarefas, é necessária a adição de comportamentos (*behaviour*) no agente. Os comportamentos representam uma tarefa que um agente pode realizar. Os comportamentos são implementados através de classes estendidas da superclasse *jade.core.behaviours.Behaviour*. Para que o agente execute um comportamento, este deve ser adicionado no agente. Isto é feito pelo método *addBehaviour()* de duas formas: dentro do método *setup()* do agente ou através de outro comportamento.

Cada classe que estende a superclasse *Behaviour* deve implementar dois métodos abstratos: *action()* e *done()*. O primeiro define as operações que serão realizadas quando o comportamento está em execução, e o segundo é um método que retorna um valor booleano e indica se o comportamento finalizou ou não.

Um agente pode executar vários comportamentos concorrentemente. Entretanto é importante ter em mente que o agendamento de comportamentos em um agente não é preemptivo, mas cooperativo. Isso significa que quando um comportamento é agendado para execução, seu método *action()* é chamado, e ele executará até seu retorno. Desta forma, é o programador que define quando um agente troca de uma execução de um comportamento para a execução de um outro. Embora isso crie alguma dificuldade na implementação de agentes, à primeira vista, pode trazer várias vantagens, segundo Bellifemine et al. (2007):

- Permite uma única *thread* Java por agente.
- Provê um desempenho melhor, uma vez que troca de comportamento no agente é mais rápido que troca de *thread* Java.
- Elimina o problema de sincronização entre comportamentos concorrentes acessando os mesmos recursos, uma vez que todos os comportamentos são executados na mesma *thread* Java.

A Figura A.5 mostra o ciclo de vida de um agente JADE. Note que enquanto o agente não foi deletado, os seus comportamentos são executados um a um, podendo finalizar ou não (a finalização do comportamento é verificada pelo método *done()*). Quando o agente é finalmente morto, ou seja, o método *doDelete()* foi chamado, o conteúdo então do método *takeDown()* é executado.

A seguir, apresentamos um exemplo simples de um agente com um comportamento de escuta de mensagem. O agente recebe e imprime todas as mensagens. Se uma mensagem chegar com o conteúdo *desligar*, o comportamento é encerrado, e o método *doDelete()* é

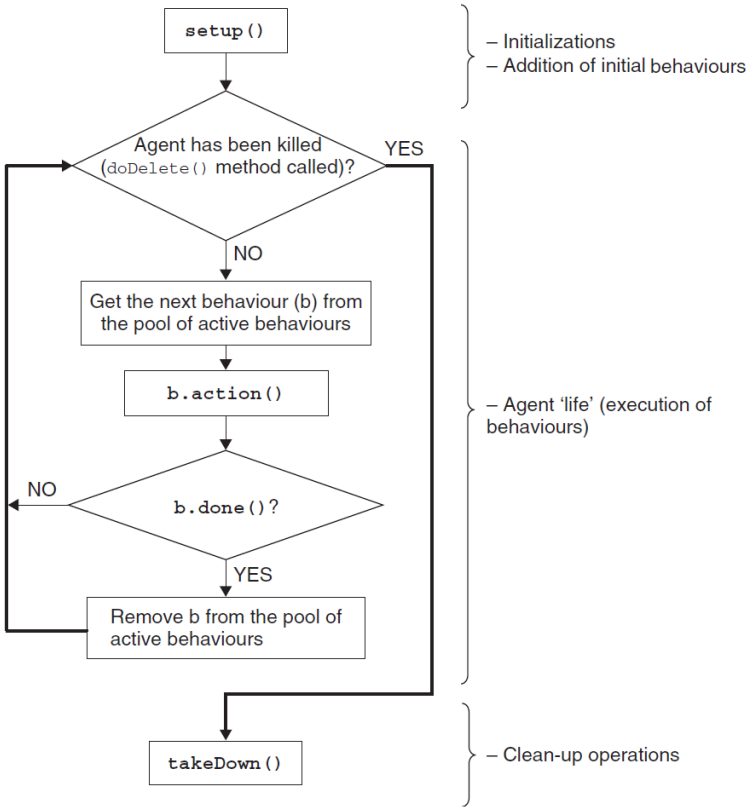


Figura A.5: Caminho de execução de um agente (Bellifemine et al., 2007)

chamado para finalizar o agente. Note que na linha 24 do código, o método *block()* é chamado. JADE recomenda a utilização deste método nos comportamentos de recebimento de mensagem para que o mesmo, após a primeira checagem de recebimento de mensagem, seja colocado na lista de comportamentos bloqueados até que chegue uma nova mensagem ou o método *restart()* seja chamado explicitamente. Isto é feito para evitar o consumo desnecessário de recursos de processamento.

```
package unb.agmi.agents;

import jade.core.Agent;
import jade.core.behaviours.Behaviour;
import jade.lang.acl.ACLMessage;
public class AgentExample extends Agent {

  @Override
  protected void setup() {
    System.out.println("Iniciar a escuta...");
    addBehaviour(new Behaviour() {
      boolean desligar = false;

      @Override
      public void action() {
        ACLMessage msg = myAgent.receive();
        if (msg != null) {
          System.out.println(msg.getContent());
          if (msg.getContent().equals("desligar")) {
```

```
20              desligar = true;
              }
22          }
            block();
24        }

26        @Override
          public boolean done() {
28          if (desligar) {
              myAgent.doDelete();
30            return true;
            }
32          return false;
          }
34      });
    }
36
    @Override
38  protected void takeDown() {
      System.out.println("Hora de ir embora...");
40  }
}
```

Observe que após a mensagem de conteúdo "desligar" ser recebida, quando o método *done()* é executado, o método *doDelete()* do agente é invocado para finalização do agente (linha 29 do código). Na Figura A.5 pode-se observar que após o método *done()* do comportamento retornar verdadeiro, o comportamento é retirado da fila de comportamentos, e então é verificado se o método *doDelete()* do agente foi invocado. Como este método foi chamado, o passo seguinte é a execução das últimas ações do agente no método *takeDown()*, finalizando-o.

Vários tipos de comportamentos são fornecidos por JADE, tais como comportamentos cíclicos, compostos, comportamento de execução única, entre outros. Para saber mais sobre o desenvolvimento de agentes utilizando a plataforma JADE, vide `http://jade.tilab.com/doc/`.

Neste capítulo apresentamos os principais fundamentos teóricos utilizados neste trabalho de pesquisa, incluindo MD, DCBD e SMA. Passaremos a apresentar o contexto de aplicação de pesquisa na área de auditoria governamental, bem como apresentaremos alguns trabalhos correlatos.