

UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE DE LA MAÎTRISE EN MATHÉMATIQUES ET
INFORMATIQUE APPLIQUÉES

PAR
MICHEL CHAREST

**INTELLIGENT DATA MINING ASSISTANCE VIA CASE-BASED REASONING AND
A FORMAL ONTOLOGY**

JANVIER 2007

Université du Québec à Trois-Rivières

Service de la bibliothèque

Avertissement

L'auteur de ce mémoire ou de cette thèse a autorisé l'Université du Québec à Trois-Rivières à diffuser, à des fins non lucratives, une copie de son mémoire ou de sa thèse.

Cette diffusion n'entraîne pas une renonciation de la part de l'auteur à ses droits de propriété intellectuelle, incluant le droit d'auteur, sur ce mémoire ou cette thèse. Notamment, la reproduction ou la publication de la totalité ou d'une partie importante de ce mémoire ou de cette thèse requiert son autorisation.

Sommaire

Ce mémoire se veut un résumé de projet de maîtrise que nous avons développé au cours des deux dernières années. Particulièrement, ce projet s'est conduit dans le cadre d'une équipe de chercheurs visant à mieux intégrer la prise de décisions et le forage de données (« data mining »).

De nos jours les gestionnaires de divers domaines (p.ex. : gestion, médecine, génie, etc.) doivent prendre des décisions éclairées sur des problèmes stratégiques, c'est-à-dire des problèmes suffisamment complexes pour avoir recours à des méthodes analytiques pour les résoudre (p.ex. : prévision, modélisation, corrélation). Étant donné que le forage est devenu un domaine très spécialisé, offrant une panoplie de techniques de divers sous-domaines tels que l'apprentissage automatisé et les statistiques, la mise en oeuvre efficace d'une activité de forage de données nécessite des connaissances pointues et des décisions appropriées sur un bon nombre de techniques spécialisées (p.ex. : le nettoyage des données, la transformation des attributs, le choix d'algorithme et de paramètres, les méthodes d'évaluation, etc.).

D'autre part, il existe un grand choix de méthodes et d'outils pour effectuer le forage de données, mais ceux-ci offrent peu de soutien « intelligent ». Par exemple, peu d'outils permettent d'effectuer de la gestion et de la réutilisation de connaissances aux fins du forage de données pour les non-spécialistes. De plus, de nos jours les chercheurs focalisent leurs efforts sur des techniques de forage très pointues, plutôt que sur les aspects stratégiques, méthodologiques et épistémologiques du forage de données. Ainsi, ce projet a donc consisté de développer un cadre théorique, conceptuel et technologique pour la réalisation d'un « assistant » intelligent pour le forage de données pour les preneurs de décision au sens général. Particulièrement, on tentera de vérifier si l'utilisation d'un système de raisonnement à base de cas et autres techniques de l'intelligence artificielle permettront de supporter la réutilisation de connaissances aux fins du data mining et de la prise de décision.

Abstract

The following work is the result of two years of intensive research and prototyping during the course of a graduate master's degree. In particular, this project was developed in collaboration with a group of researchers with the common goal of better integrating data mining with decision support.

Nowadays, decision makers in very competitive and diverse business sectors (i.e. management, medical, engineering, etc.) must make informed decisions about strategic problems. Such problems are often complex enough to warrant the use of sophisticated analytical methods for their resolution (i.e. forecasting, modeling, regression, etc.). Having borrowed from the myriad of techniques and models available from the areas of machine learning and statistics, data mining has become a very specialized field. Consequently, the effective application of DM is littered with many difficult and technical decisions (i.e. data cleansing, feature transformations, algorithm and parameter selection, model evaluation).

Most data mining products provide a large number of models and tools, but few provide "intelligent" assistance. For instance, few DM tool vendors provide non-specialist data miners with the ability to manage and reuse useful DM knowledge (i.e. how to perform data cleansing and feature transformations, etc.). Moreover, research seems to be based on utterly specialized DM techniques, rather than focusing on strategic, methodological, and even epistemological aspects of DM. As a result, it has been our goal to put forward a theoretical, conceptual, and technological framework for the realization of an intelligent data mining assistant, capable of empowering non-specialist data miners and decision makers. Specifically, we attempt to verify if the use of a case-based reasoning system and other artificial intelligence techniques will provide an adequate environment for the reuse of data mining knowledge with the intention of better supporting the decision making process.

Résumé

Introduction

La mise en œuvre efficace d'une activité de forage de données nécessite des connaissances pointues et des décisions appropriées sur un bon nombre de techniques spécialisées (p.ex. : le nettoyage de données, la transformation des attributs, le choix d'algorithme et de paramètres, les méthodes d'évaluations, etc.). De nos jours, il existe un grand choix de méthodes, de modèles et d'outils pour effectuer le forage de données, mais peu de soutien « intelligent » pour les non experts. Ainsi, suite à des recherches, nous avons réalisé un assistant pour le forage de données, basé sur le raisonnement à base de cas et une ontologie formelle, capable d'assister les non-spécialistes lors de leur démarche d'activités de forage de données.

Afin de demeurer efficace les preneurs de décisions ont fréquemment recourt aux techniques de forage de données pour lutter contre l'accroissement incessant d'informations suite aux opérations quotidiennes de leur entreprise. Malgré le fait que le data mining semble très prometteur pour assister à la découverte de « connaissance », l'application efficace du processus de forage de données comprend à la fois de grands défis et difficultés. Par exemple, les recherches actuelles menées sur le forage de données sont basées sur l'application de techniques très spécialisées (p.ex. : les statistiques, l'apprentissage automatique et les bases de la théorie de l'information), or la recherche portant sur des thèmes méthodologiques, stratégiques ou épistémologiques se font plutôt rare. D'autre part, sur le plan pratique très peu d'entreprises utilisent des méthodes de gestion des connaissances sur l'application pratique du forage de données (p. ex: une mémoire institutionnelle). Par ce fait, les anecdotes de réalisations fructueuses utilisant le forage de données se font rares. De plus, malgré le fait que les méthodologies fréquemment employées pour le forage (telle que la méthodologie CRISP-DM) offrent des consignes générales pour guider les utilisateurs dans leurs démarches, les non-spécialistes ont plutôt besoin de suggestions et d'explications dans des contextes précis lors de la démarche du processus. Autrement dit, il n'est pas suffisant de dire « quoi » un utilisateur doit faire, mais il est

plutôt important de dire « comment » et à quels instants on doit appliquer une telle technique, méthode ou vérification lors d'une activité de forage. Enfin, la majorité des assistants réalisés au fil des années ont focalisés uniquement à supporter le bon choix de modèle pour une activité de forage de données. Malgré que cette étape soit importante, elle ne peut assurer le succès d'une activité de forage de données. Ainsi, un assistant intelligent devrait offrir un support tout au long de l'application du processus de forage (p.ex. : assister l'analyse et la préparation des données, l'évaluation de modèles).

Objectifs du travail de recherche

Suite à une étude approfondie des problématiques de ce domaine, nous nous sommes fixés les objectifs de recherche suivants :

- 1) **Supporter les non-spécialistes** – Assister les analystes non experts du forage de données en considérant particulièrement leur niveau de connaissances du forage de données.
- 2) **La réutilisation de connaissances** – Encourager la réutilisation d'expériences antérieures de data mining sous la forme d'une base de connaissance ou de mémoire institutionnelle.
- 3) **Un soutien holistique** – Apporter un soutien au-delà de l'assistance au choix de modèle, mais plutôt un support qui comprend les étapes majeures telles que la préparation de données, la modélisation et l'évaluation des modèles.
- 4) **Utiliser des connaissances approfondies** – offrir des connaissances sous la forme de suggestions, heuristiques et réponses automatiques pour assister l'utilisateur à prendre des décisions lors de la démarche du processus de forage de données.

En résumé, nous avons tenté de rendre le forage de données plus accessible et facile pour les non-spécialistes de ce domaine en proposant un cadre théorique, conceptuel et technologique pour la réalisation d'un assistant intelligent pour le forage de données.

Plus particulièrement, nous avons vérifié si la combinaison « synergique » d'un système de raisonnement à base de cas (RBC) et d'une ontologie formelle peut supporter de façon convenable les non experts pratiquant le forage de données.

La méthodologie utilisée

Dans un premier temps, nous avons effectué une analyse approfondie de l'état de l'art sur les assistants de forage de données, ainsi que les systèmes d'aide à la décision pertinents à celui-ci. Ceci nous a permis de bien cibler et de définir les problématiques. En fait, ceci nous a permis d'élaborer nos objectifs de recherches tels qu'ils le sont énoncés ci-dessus. Deuxièmement, nous avons effectué une analyse approfondie de l'état de l'art de plusieurs domaines sous-jacents, le forage de données et les systèmes de prise de décisions. Par exemple, nous avons enquêté sur les avancements réalisés au niveau des processus de forage de données, le méta-apprentissage et la caractérisation de données. Ensuite, nous avons examiné les divers modes de représentation de connaissances (et méthodes de raisonnement respectives) tels que le raisonnement à base de cas et les ontologies formelles basées sur la logique de description.

Particulièrement, nous avons évalué un bon ensemble de cadres et de systèmes de raisonnement à base de cas dans les milieux académiques et professionnels. Ceci nous a permis de conclure qu'il était préférable de concevoir et de réaliser notre propre système de raisonnement à base de cas. La première étape importante pour la réalisation de notre système RBC a consisté à définir une représentation d'un cas de forage de données, c'est-à-dire un ensemble de caractéristiques représentatives d'une activité de data mining. Pour ce faire, nous avons examiné attentivement le processus de forage de données CRISP-DM. Malgré que celui-ci est représenté en utilisant le langage naturel (p.ex. : anglais), nous avons pu définir une représentation d'un cas de forage comportant un ensemble de 66 caractéristiques des 5 phases principales du processus CRISP-DM (p.ex. : les besoins d'affaires, la compréhension des données, la préparation de données, la modélisation, et l'évaluation du processus). Ayant conçu une représentation abstraite d'un cas de data mining, nous avons ensuite réalisé une composante pour faire la comparaison de cas de forage de données, c'est-à-dire la

réalisation d'une mesure d'appariement globale (et les mesures de similarités locales sous-jacentes) nous permettant d'effectuer une comparaison quantitative entre deux cas de forage de données. Ainsi, un usager ayant stocké des activités de forage antérieurement dans notre base de cas est en mesure de repérer des cas « similaires » à son problème de forage de données actuel.

À cette étape de nos initiatives, ayant conçu la base de notre assistant de forage de donnée en utilisant un RBC, nous avons effectué de premiers essais. Ces essais nous ont permis de constater deux lacunes importantes à notre système :

- a) Quand un usager utilise notre système pour repérer un ensemble de cas antérieurs et similaires à son problème actuel, il n'est pas évident pour l'utilisateur de déduire quel « cas de base » est le meilleur choix pour débiter le processus d'adaptation et éventuellement résoudre son cas actuel de forage de données.
- b) Ayant choisi un cas relativement similaire pour résoudre le problème actuel, il n'est pas toujours évident pour l'utilisateur de savoir quelles informations dans ce cas de base sont utiles et pertinentes au problème actuel qu'il doit résoudre (adaptation d'un cas de base au cas de forage actuel).

Ainsi, pour résoudre le problème de sélection de cas de base reporté par le système de raisonnement à base de cas, nous avons proposé la réalisation d'une mesure supplémentaire basée sur la théorie de l'utilité. Cette nouvelle mesure sert à donner à l'utilisateur un indice du niveau de « qualité » ou capacité de résolution d'un cas similaire. Ainsi, l'utilisateur peut maintenant faire un choix final du cas de base à utiliser basé sur un compromis (ou équilibre) entre le niveau de similarité d'un cas et le niveau potentiel d'utilité ou d'adaptabilité de ce cas par rapport au cas de forage de données du problème à résoudre.

D'autre part, pour résoudre le second problème d'adaptation d'un cas de base, c'est-à-dire offrir des suggestions précises pour aider l'utilisateur à modifier les caractéristiques pertinentes d'un cas similaire, nous avons constaté le besoin d'une base de connaissances supplémentaire pour offrir cette aide. En fait, au début, puisque les ontologies basées sur la logique de description offrent naturellement la représentation

de connaissance déclarative, nous avons tenté d'utiliser celles-ci (et les techniques de raisonnement de la logique de description) pour résoudre ce problème. Mais, suite à des essais, nous avons rapidement constaté qu'il était nécessaire d'ajouter une base de connaissance supplémentaire contenant des connaissances procédurales (des connaissances à base de règles). Ainsi, par la suite, nous avons effectué des enquêtes sur des cadres ontologiques offrant la possibilité de représenter à la fois des concepts déclaratif et procéduraux. Enfin, malgré que cette avenue est actuellement un sujet de recherche à ses balbutiements, nous avons réussi à intégrer des connaissances approfondies sur le data mining sous la forme de règles et concepts dans notre ontologie formelle (suite à une activité d'ingénierie et formalisation de certaines connaissances de forage de données).

Finalement, la résolution de ces problèmes fondamentaux nous a permis de mettre en œuvre un assistant intelligent pour le forage de données. Ensuite, nous avons procédé à l'évaluation de notre système tel qu'indiqué dans la section suivante.

Les résultats obtenus

Ayant à la fois défini un ensemble de cas de forage de données « noyau » afin de rendre fonctionnel notre système RBC et codé un premier ensemble de connaissances approfondies (sous la forme de concepts et règles) dans notre ontologie formelle, nous avons procédé à la réalisation de quelques activités de forages de données. Enfin, nous avons fait une évaluation comparative des suggestions fournies par notre système intelligent à ceux d'un expert humain (voir Section 5 pour plus de détails).

Conclusions

Nous avons réalisé un système intelligent pour le forage de données basées sur la synergie d'un système de raisonnement à base de cas et d'une ontologie formelle. La première composante (RBC) permet à l'utilisateur de faire évoluer une sorte de mémoire institutionnelle de cas de forage de données (au fur et à la mesure qu'il résout des nouveaux cas) qui permet de facilement repérer des cas similaires antérieurement résolus pour avoir un premier aperçu sur la résolution du problème actuel.

D'autre part, la composante ontologique de notre système contenant des règles et suggestions textuelles, permet d'offrir des connaissances pointues et précises à un usager lorsqu'il effectue une activité de forage (suite au raisonnement qu'offre un moteur de raisonnement à base de règles). C'est-à-dire, le système intelligent offre des suggestions précises à l'utilisateur pendant que celui-ci procède à la résolution de son problème de forage en modifiant des caractéristiques du cas de base.

De plus, ces deux modes de représentations de connaissance complémentaires permettent aux non-spécialistes de profiter d'une assistance holistique sur l'activité de forage de données. Enfin, il est très important de mentionner que nos objectifs de recherches ont été particulièrement abordés dans la perspective d'apporter un soutien aux non-spécialistes du forage de données, c'est-à-dire les preneurs de décision au sens général du terme.

Remerciements

Sans la générosité et contributions de plusieurs personnes, ce projet ne se serait fort probablement jamais réalisé.

Premièrement, je tiens à remercier **Hélène, Thérèse et Claude Grandmont** pour leur soutien et encouragement constant tout au long du cheminement de mon programme de maîtrise. Particulièrement à ma douce moitié Hélène, je te remercie profondément pour ta patience et compréhension lorsque j'ai dû faire des travaux au lieu de profiter des sentiers de nature et du tennis avec toi. D'autre part, je serai éternellement reconnaissant à Claude et Thérèse pour toute leur générosité et gentillesse lors de mes nombreux séjours à Trois-Rivières.

Je tiens également à remercier **Sylvain Delisle**, mon directeur de recherche et en quelque sorte un « éclairer » tout au long de mon expédition dans ce merveilleux univers de la recherche scientifique et informatique. Je te remercie sincèrement pour ta disponibilité, générosité et persévérance lors de nos rencontres pour éclaircir les divers points subtils sur le forage de données, les ontologies, etc.

Enfin, j'aimerais remercier **Patrice Gagnon et Yves Côté** pour leur soutien et encouragement par l'entremise de nos fréquentes discussions philosophiques et même parfois banales, lors de nos pauses café à la Chasse !

Table of Contents

Sommaire	i
Abstract.....	ii
Résumé	iii
Remerciements.....	ix
Table of Contents.....	x
List of Figures	xiv
List of Tables.....	xvi
Abbreviations	xvii
CHAPTER 1 Introduction	1
1.1 Data Mining and Decision Support.....	1
1.2 The Decision Support and Data Mining Paradox.....	2
1.3 Research Objectives – An Executive Summary.....	4
1.4 Content and Scope of Document	5
CHAPTER 2 Problem Definition.....	6
2.1 The Challenges of DM Assistance.....	6
2.1.1 The Inherent Complexity of Data Mining.....	6
2.1.2 Support for the Non-Expert Data Miner	7
2.1.3 Fostering Knowledge Reuse	8
2.1.4 Beyond Model Selection Support	9
2.1.5 A Need for Detailed DM Knowledge.....	9
2.2 Knowledge Representation and Reasoning Challenges.....	10
2.2.1 Case-Based Reasoning	10
2.2.1.1 Limited Availability of CBR Frameworks	12
2.2.1.2 The Need for a KI-CBR Framework.....	12
2.2.2 OWL DL Ontologies	13
2.3 Data Mining Without Assistance – A Simple Example	14
2.4 A Formal Research Objectives Statement.....	18
2.4.1 The UQTR Decision Support Department - A Real Case Study	19
CHAPTER 3 State of the Art	20

3.1	Decision Support Systems	20
3.1.1	The Anatomy of a DSS.....	21
3.1.2	Applications Areas for DSS	22
3.1.3	Knowledge Management Systems	22
3.1.4	Data Mining Assistants.....	23
3.2	The CRISP-DM Process	24
3.3	Meta-Learning.....	27
3.3.1	Dataset Characterization.....	28
3.3.1.1	General, Statistical and Information-Theoretic Measures	28
3.3.1.2	Model-Based Characterization	29
3.3.1.3	Landmarking	29
3.3.2	Model Selection Assistance.....	29
3.3.2.1	Learning at the Meta-Level.....	30
3.3.2.2	Model Ranking Methods.....	30
3.3.3	The Learning-to-Learn Paradigm	31
3.3.4	Inductive Transfer	31
3.4	Case-Based Reasoning	32
3.4.1	Definition.....	32
3.4.2	Some Advantages of Using CBR	34
3.4.3	Common Applications of CBR.....	35
3.4.4	Retrieval and Similarity Assessment Methods.....	36
3.4.4.1	Surface Similarity Approaches.....	36
3.4.4.2	Structural Similarity Approaches.....	37
3.4.5	Reuse and Revision Strategies	38
3.4.6	Retention and Maintenance Strategies.....	40
3.4.7	An Evaluation of Available CBR Frameworks.....	41
3.5	Ontologies.....	43
3.5.1	Definition.....	43
3.5.2	Some Advantages of Using Ontologies	44
3.5.3	Common Applications of Ontologies.....	44
3.5.4	Ontology Markup Languages	46
3.5.5	An Evaluation of Rule-Supported Ontology Frameworks.....	48

3.5.6	Ontologies and Rule-Based Reasoning.....	50
3.6	The Knowledge Engineering Process.....	51
3.6.1	Knowledge Engineering Model Types.....	51
3.6.2	A Knowledge Engineering Process Model.....	53
3.7	Knowledge Intensive CBR Systems Revisited.....	54
3.7.1	CBR and Rule-Based Reasoning.....	55
3.7.2	CBR and Ontologies.....	55
CHAPTER 4 The Proposed Intelligent Data Mining Assistant.....		57
4.1	Grounded in Meta-Learning.....	58
4.2	A CRISP-DM Driven Process.....	59
4.3	CBR at the Core.....	61
4.3.1	The Key Components of a CBR System.....	61
4.3.2	Case Vocabulary Elicitation.....	63
4.3.3	Problem Characterization and Feature Indexes.....	65
4.3.3.1	Data Characterization and Incomplete Data Sets.....	68
4.3.4	Similarity and Utility-Oriented Retrieval.....	69
4.3.4.1	The Similarity Measures.....	69
4.3.4.2	The Utility Measures.....	72
4.3.5	Reuse and Revision Strategy.....	75
4.3.6	Case Retention.....	76
4.3.7	Eliciting Seed Cases.....	76
4.3.8	Case Maintenance Issues.....	77
4.4	The OWL-DL Ontology Component.....	78
4.4.1	Ontology Knowledge Elicitation.....	79
4.4.1.1	Concepts, Properties and Individuals Elicitation.....	80
4.4.1.2	SWRL Rules Elicitation.....	82
4.4.2	The Rule-Based Inference Component.....	84
4.4.3	Ontology-Guided Intelligent DM Assistance.....	85
4.4.3.1	The Initial Bootstrap Advice.....	88
4.4.3.2	Terminological Definitions.....	88
4.4.3.3	Recommendations and Heuristics.....	89
4.4.3.4	Scope of Detailed DM Knowledge.....	89

4.4.3.5	A Note on Rule Opacity.....	90
4.5	Intelligent DM Assistant System Overview	91
4.5.1	Implementation Issues	92
CHAPTER 5 Tests, Results and Validation		94
5.1	A Quick Tour.....	94
5.2	Assisted Data Mining Problems.....	100
5.2.1	A Classification Example.....	100
5.2.2	A Class Imbalance Example.....	105
5.2.3	A Feature Reduction Example.....	107
5.2.4	A Regression Example.....	108
5.3	A Brief System Evaluation.....	110
CHAPTER 6 Future Directions.....		113
6.1	Improving Problem Characterization	113
6.2	Beyond Classification Support.....	114
6.3	CBR to Ontology Knowledge Promotion.....	115
6.4	Case Maintenance Certification.....	115
6.5	Leveraging DL Reasoning during Case Adaptation.....	116
6.6	Ontology-Based Visual Explorer.....	116
6.7	Improving the Case Adaptation Interface.....	117
6.8	Ontological DM Body of Knowledge	117
6.9	Integration with Data Warehousing.....	118
6.10	Supporting the Deployment Phase.....	118
6.11	Overall System Performance Using Utility	119
CHAPTER 7 Conclusions.....		120
Appendix A - Description Logic Classification.....		123
Appendix B - DM Problem Characteristics Details.....		124
Appendix C - List of Data Mining Case Attributes.....		126
Appendix D - List of Elicited SWRL Rules		128
References.....		133

List of Figures

Figure 1 The Decision Support and Data Mining Paradox	3
Figure 2 Oracle Data Miner Wizard – Handling Outliers (source [5]).....	8
Figure 3 Some Challenges Associated With Exploiting the CBR Paradigm	11
Figure 4 Selecting a Feature Transformation Filter Using Weka (source [26])	15
Figure 5 Selecting a Machine Learning Algorithm Using Weka (source [26]).....	16
Figure 6 Setting Training Parameters Using Weka (source [26])	17
Figure 7 Examining the Model Evaluation Parameters Using Weka (source [26]).....	18
Figure 8 - A Schematic View of a DSS (source [27]).....	21
Figure 9 The CRISP-DM Process Model Life-Cycle (source [37]).....	25
Figure 10 The CRISP-DM Data Preparation Phase Flowchart (source [37]).....	27
Figure 11 Learning at the Meta-Level	30
Figure 12 Model Ranking Example for a given DM Problem (source [47])	31
Figure 13 The Case-Based Reasoning Cycle (source [51])	33
Figure 14 A Classification Hierarchy of CBR Applications (source [17]).....	35
Figure 15 OWL DL Descriptions, Data Ranges Properties and Values (source [20]) ...	48
Figure 16 The Common KADS Knowledge Engineering Process (source [95]).....	54
Figure 17 Architectural Overview of Intelligent Data Mining Assistant (source [2]).....	57
Figure 18 A Data Mining Meta-Learning Problem (source [104])	58
Figure 19 The View of CBR via Knowledge Containers (source [51])	62
Figure 20 The Distribution of Feature Index Types	68
Figure 21 Index Means/Modes from Statistical Analysis of Domain Data.....	77
Figure 22 Data Preparation Knowledge Elicitation Taxonomy.....	80
Figure 23 Data Mining Models Taxonomy.....	82
Figure 24 Conceptual View of the Reasoning Cycle	85
Figure 25 A Conceptual View of CBR and Ontology Synergy Using SWRL Rules.....	86
Figure 26 Intelligent Data Mining Assistant System Overview (source [113]).....	91
Figure 27 A Deployment Diagram of the DM Assistant	93
Figure 28 Intelligent Data Mining Assistant Main Interface.....	94
Figure 29 Data Source Specification and Characterization Step.....	95
Figure 30 Bootstrap Information and the Retrieval Step.....	96

Figure 31 Selecting a Suitable DM Basis Case.....	97
Figure 32 Reviewing the Basis Case and the Reuse Step	98
Figure 33 Recommendations and the Case Adaptation Step.....	99
Figure 34 Context Sensitive Support for CRISP-DM Terminology	99
Figure 35 Selecting a Basis Case for a Classification Example	101
Figure 36 Existing Case Recommendations for the Data Preparation Phase.....	101
Figure 37 Ontology-Driven Recommendation for the Data Preparation Phase	102
Figure 38 Existing Case Recommendations for Data Modeling Phase	102
Figure 39 Ontology-driven Recommendations for Data Modeling Phase	103
Figure 40 Selecting a Basis Case for a Class Imbalance Example	105
Figure 41 CBR to Ontology Knowledge Promotion	116

List of Tables

- Table 1 Summary of Evaluated CBR Frameworks..... 42
- Table 2 Summary of Ontology and Rule Reasoning Frameworks..... 49
- Table 3 Local Similarity Measures and Weights..... 71
- Table 4 Local Utility Measures and Weights 73
- Table 5 Similarity and Utility-Oriented Retrieval Example 74
- Table 6 Summary of Recommendations for a Classification Example 104
- Table 7 Summary of Recommendations for a Class Imbalance Example 106
- Table 8 Summary of Recommendations for a Feature Reduction Example 107
- Table 9 Summary of Recommendations for a Regression Example 108
- Table 10 Qualitative System Evaluation 111
- Table 11 Description Logic Classification Symbols 123
- Table 12 Problem Characteristics (Feature Indexes) 124
- Table 13 Data Mining Case Attributes 126

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
ARFF	Attribute Relation File Format
CBR	Case-Based Reasoning
CRISP-DM	Cross Industry Standard Process for Data Mining
DBMS	Database Management System
DL	Description Logic
DM	Data Mining
DP	Data Preparation
DU	Data Understanding
DW	Data Warehouse
DSS	Decision Support System
FOL	First Order Logic
GUI	Graphical User Interface
GSM	Global Similarity Measure
GUM	Global Utility Measure
IBL	Instance-Based Learning
JESS	Java Expert System Shell
KBS	Knowledge-Based Systems
KDD	Knowledge Discovery in Databases
KMS	Knowledge Management System
KI-CBR	Knowledge Intensive Case-Based Reasoning
LSM	Local Similarity Measure
LUM	Local Utility Measure
ML	Machine Learning
OWL	Ontology Web Language
PCA	Principal Component Analysis

PSM	Problem Solving Method
RDBMS	Relational Database Management System
RBES	Rule Based Expert System
RDF	Resource Description Framework
SAS	Statistical Analysis System
SMO	Sequential Minimal Optimization
SVM	Support Vector Machine
SPSS	Statistical Package for the Social Sciences
SVD	Singular Value Decomposition
SWRL	Semantic Web Rule Language
UQTR	Université du Québec à Trois-Rivières
WEKA	Wakaito Environment for Knowledge Acquisition
XML	Extensible Markup Language
XSLT	XML Stylesheet Language Transformation

CHAPTER 1

Introduction

This chapter begins by introducing the fundamental premise and « raison d'être » for our research endeavor – the fields of data mining and decision support systems. Subsequently, in order to gain a better appreciation of the problems that plague decision makers wishing to leverage data mining technology, we present the “*Decision Support and Data Mining Paradox*”, a conceptual view we have coined early on during our research in order to remain focused and resolve our research objectives. Last, we briefly state our research objectives and provide a concise overview of the content and scope of this memoire.

1.1 Data Mining and Decision Support

Data Mining is the non-trivial extraction of implicit, potentially useful information from data. The acquired knowledge is usually obtained from the use of a generated model such as a rule-set, decision tree or regression formula. The ultimate goal of data mining is to discover knowledge. Over the past decade, the field of data mining has evolved substantially from the wide breath of concepts and methods borrowed from areas such as statistics, database management, machine learning, soft computing and data visualization.

Data mining has traditionally been defined as an integral part of the KDD process. The scope of KDD covers the overall process of converting raw data into useful information or knowledge. This process consists of a series of steps:

- **Pre-processing** – Transform the raw input data into a form that is more appropriate for data modeling (i.e. data cleansing, feature transformation, examples reduction, etc.).

- **Modeling or Data Mining** – Using various statistical and machine learning techniques to analyze the raw data and produce a “generalized” model (i.e. decision tree, regression equation, neural network, etc.).
- **Post-processing** – Used to ensure that only valid and useful results are integrated into the decision support system. (i.e. model comparisons, hypothesis testing, pattern interpretation, deployment).

However, with the widespread use of DM over the past decade, the boundaries between DM and KDD are not so clearly delineated anymore. It is not uncommon for DM processes to encompass both pre-processing and post-processing steps, since these are inextricably dependent on the model generation step. Hence, for practical purposes the following work will use KDD and DM interchangeably. Although we shall be addressing the concepts of decision support and DSS more thoroughly in Section 3, the following provides a brief explanation in order to satisfy the current discussion. A DSS refers to a system which assists decision makers by combining data, tools and sophisticated analytical models (and at times knowledge-based sub-systems) into a powerful application that can support the resolution of strategic¹ decision problems within an organization.

1.2 The Decision Support and Data Mining Paradox

The main purpose of a decision support system is to increase the effectiveness of decision makers for resolving complex, non-structured problems. From another perspective, data mining methods and tools hold the promise of facilitating the decision maker's life by extracting hidden information from data in order to support decision making. As a result, nowadays decision makers (i.e. non-specialist data miners) must not only contend with the complexities of their data, they must also manage the inherent complexity associated with effectively applying the available “arsenal” of data mining

¹ A strategic problem is a complex task for which no well-defined procedure exists for resolving it (but rather requires a sophisticated analytical, probabilistic and/or knowledge-based method for its resolution), while a non-strategic or structured problem is one for which a well-defined procedure is available.

tools, methods and algorithms. Frequently, decision makers are stumped with having to make difficult DM related decisions, let alone the eventual business decisions that will ensue from a DM effort. In other words, in order for decision makers to effectively profit from DM technology, the DM technology must in return make use of intrinsic application domain knowledge.

This paradox or contentious situation is metaphorically demonstrated in Figure 1. Nonetheless, if a bridging mechanism such as an intelligent data mining assistant can be implemented to bring together such disparate yet complementary disciplines, effective results can be obtained for the decision maker or novice data miner. The paradox clearly stems from poor data mining and decision support integration and provides the fundamental premise upon which our research is founded and documented herein.

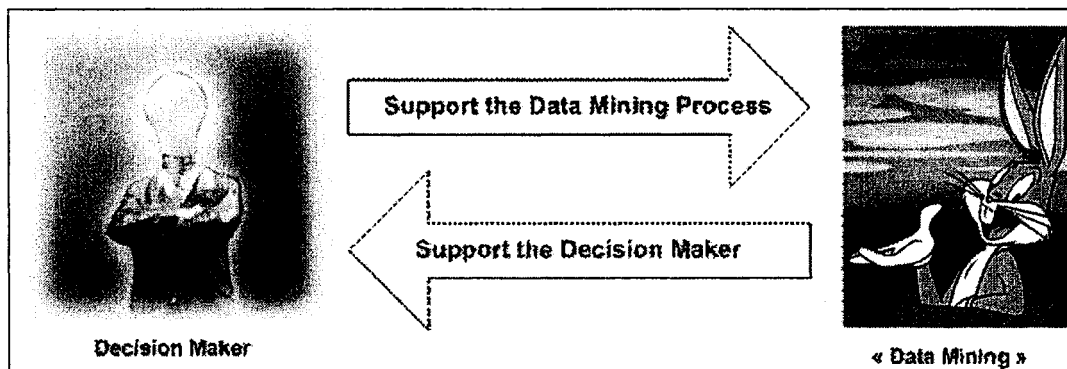


Figure 1 The Decision Support and Data Mining Paradox²

It is worth mentioning that our appeal to a decision maker is not restricted to senior management personnel or a business person in the traditional sense. Our view of decision making and decision support is rather applicable to personnel within all levels of an organization or business sector (i.e. government, academic, business, engineering, medical, etc.). The key requirement is that a person should need to assume a decision making role and wish to leverage DM technology in order to better achieve her decision making goals or objectives.

² © The Bugs Bunny cartoon is a copyright of Looney Tunes, Warner Bros.

1.3 Research Objectives – An Executive Summary

Since we shall be revisiting DM challenges and objectives more formally in Section 2, the following presents an executive summary of our targeted research objectives. Although data mining does promise to uncover valuable and useful knowledge, the effective application of data mining still faces some very serious challenges. As a result, we have attempted to address the following key challenges ([27], [23]):

- **Support for the Non-Expert Data Miner** – Current DM research is based on very specialized techniques (statistics, machine learning, information theory, etc.), whereas DM research on strategic, methodological, and epistemological aspects are rare.
- **Fostering Knowledge Reuse** - Current DM processes make very little use of existing corporate knowledge. Consequently, DM is more tedious than necessary and can tend to produce already known information.
- **Beyond Model Selection Support** - Previous research efforts into DM assistants have primarily focused on providing a user with model selection support. Novice data miners require a more holistic approach that provides assistance throughout the entire data mining process (i.e. pre-processing and post-processing).
- **A Need for Detailed DM Knowledge** - Existing DM methodologies provide general directives, however what a non-specialists really need are explanations recommendations on how to effectively carry out particular steps of a DM methodology.

In essence, we have attempted to make DM more accessible and effective for non-specialist data miners by proposing a theoretical, conceptual and technological framework from which we have implemented an intelligent DM assistant. More specifically, we have tested if the combined use of a CBR and formal DL ontology can

efficiently support non-specialist data miners, thereby fulfilling its requirements in addressing the aforementioned DM challenges.

1.4 Content and Scope of Document

Section 2 – Problem Definition - introduces some of the key challenges associated with providing data mining assistance for novice data miners. In addition, we present a simple data mining example and highlight the typical decisions a novice data miner must face when using commercial data mining toolkits. Last, we define and clearly state our intended research objectives.

Section 3 – State of the Art - provides basic definitions, concepts and a brief state of the art for each of the key elements that were eventually integrated within our intelligent data mining assistant implementation.

Section 4 – The Proposed Intelligent Data Mining Assistant - presents the key design considerations and issues considered during the realization of our hybrid intelligent data mining assistant. In particular, we address the meta-learning, CBR and formal ontology principles that have influenced the design of our DM assistant.

Section 5 – Test, Results and Validation - provides a quick tour of the intelligent data mining assistant. Subsequently, we examine how the data mining assistant provides recommendations and heuristics for several data mining problems. Last, we provide a brief performance comparison of the recommendations offered by our data mining assistant to those offered by a data mining professional.

Section 6 – Future Directions - presents some potentially useful and interesting future research directions from which the basis of our current work may be extended.

Section 7 – Conclusions - provides a summary of the key solution elements and benefits offered by our proposed intelligent data mining assistant.

CHAPTER 2

Problem Definition

This chapter first introduces some of the key challenges associated with providing data mining assistance for novice data miners. Second, we present some specific knowledge representation and reasoning challenges associated with implementing an “intelligent” data mining assistant. Third, in order to better appreciate the need for intelligent data mining assistance, we present a simple data mining example and highlight the typical decisions a novice data miner must face when using commercial (or academic) data mining toolkits. Last, we define and clearly state our intended research objectives.

2.1 The Challenges of DM Assistance

Although the challenges associated with providing DM assistance are numerous the following shall concentrate on some of the key challenges we have attempted to resolve throughout the course of our research.

2.1.1 The Inherent Complexity of Data Mining

Over the past several decades, the field of data mining has witnessed tremendous growth by profiting from the advancements of numerous areas (machine learning, statistics, information-theory, data-warehousing, etc.) and specialized sub-fields (i.e. data visualization, neural networks, probabilistic methods, ensemble learning, etc.) [36]. Nowadays, not only must data miners contend with the complexities of their respective fields of application (i.e. possess adequate domain knowledge to effectively interpret the data), they must also manage the inherent complexities associated with effectively using the available “arsenal” of data mining tools, methods and algorithms. In brief, without aiming to become an expert data miner, a novice user must become reasonably familiar and skilled with dealing with some of the following issues:

- a) How to effectively perform data quality verification (i.e. missing values, outliers)?
- b) How to efficiently perform the data preparation phase (i.e. normalization, discretization, binarization)?

- c) Which statistical or machine learning algorithm is most appropriate (i.e. decision tree, neural network, logistic regression, etc.)?
- d) Which training parameters are most suitable?
- e) How to deal with a potential class imbalance problem?
- f) How to deal with the curse of dimensionality?
- g) How to avoid model over-fitting?
- h) How to improve the accuracy rate (i.e. error rate)?
- i) How to evaluate the data mining effort (cross-validation, p-value, ROC-curves)?

Particularly, the fields of statistics and machine learning have produced a myriad of models and algorithms that can readily be exploited by data miners. Consequently, this profusion of algorithms has dramatically burdened the data miner with more difficult decisions that must be addressed in order to effectively apply DM to produce useful and meaningful results (i.e. most algorithms tend to offer a host of specialized parameters that can be adjusted in order to achieve better performance) [53].

2.1.2 Support for the Non-Expert Data Miner

Most commercial data mining products (i.e. Oracle Data Miner [74], SAS Enterprise Miner [89]) either do not offer any intelligent assistance (i.e. decision support) or tend to do so in the form of rudimentary “wizard-like” interfaces. These wizard-like interfaces make hard assumptions about the level of background knowledge required by a user. Giraud Carrier *et al.* [35] have further substantiated this fact during a detailed evaluation of the data mining advisor (*MetaL*). For instance, amongst the many decisions a novice data miner must make during the application of an entire data mining process, a user must frequently decide how to handle outlier values within the problem data set. Though a typical DM toolkit interface, as illustrated by Figure 2, can provide choices for handling outlier values (i.e. cut-off points, replacement values, etc.), without background knowledge a novice user can easily make poor decisions and obtain more than questionable DM results. On the other hand, with a little assistance, the handling of outlier values can successfully be mitigated by providing a recommendation such as using a dispersion diagram to visually detect and remove potential outliers (instead of blindly ignoring all outliers or eliminating all values beyond a pre-defined threshold as is

often recommended by a DM toolkit (see Figure 2). Evidently, we must also stress that the preceding recommendation should also further suggest to the user that a careful analysis to determine the potential cause of the outliers is also necessary. Otherwise, without proper interpretation, carrying out the recommendation blindly will probably yield no better a result (i.e. model error rate) than simply ignoring the outliers [118].

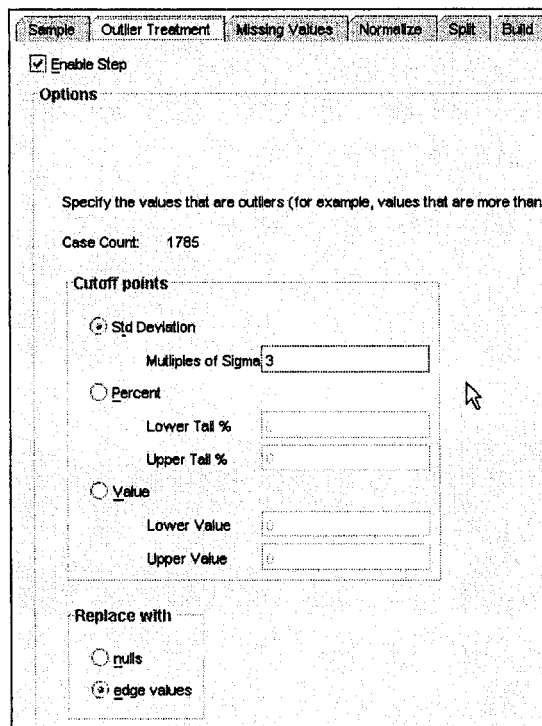


Figure 2 Oracle Data Miner Wizard – Handling Outliers (source [74])

2.1.3 Fostering Knowledge Reuse

With respect to the overall data mining process, most enterprises do not directly manage tacit DM knowledge in a form that can be effectively stored, refined and reused. Most products simply archive DM activities, but leave it up to the user to intelligently manage this knowledge. Examples of this DM knowledge are provided by the items a) to i) in the above list. As previously stated by Charest *et al.* [11], an intelligent DM assistant should possess characteristics that allow it to learn from past experience and empower the user of the system to avoid the repetition of mistakes.

2.1.4 Beyond Model Selection Support

Previous research efforts into intelligent DM assistants have primarily focused on providing a user with model selection support (*NOEMON* [52], *MetaL* [65], *AST* [59], *CAMLET* [100]). The selection of an appropriate algorithm for a given data mining task³ may be considered necessary, but is definitely not sufficient for ensuring the successful outcome of a DM project. An intelligent DM assistant implies the realization of a system that is capable of aiding a user throughout the various phases of the data mining process. Novice data miners require a data mining assistant that encourages a more holistic approach (i.e. data understanding, preparation, modeling and evaluation) towards the task of data mining.

2.1.5 A Need for Detailed DM Knowledge

As shall be elaborated upon in Section 3.1, though methodologies adequately specify the phases, tasks and activities that need to be carried out during a DM project (and corresponding inputs and outputs), these provide very little detailed knowledge for the novice miner on how (and specifically when) to actually carry out a given step. For example, the proper generation of a simple linear regression model requires that the user possess detailed knowledge for effectively carrying out a series of essential activities (i.e. verify linearity assumption, perform significance testing of the null hypothesis; verify residue normality and variance constancy) [36]. Though the specification of the CRISP-DM is effective at providing general guidelines and tips, it does not provide detailed data mining “know-how” that a novice user must possess in order to better her chances of successfully applying the data within the context of a “real-world” application. As shall be elaborated upon in Section 4, this detailed DM knowledge can most naturally be expressed in a “procedural” or rule-like form.

³ In the present context, “task” is used to generically designate an entire data mining activity (i.e. classification, regression, clustering, etc.). However, at times throughout the document, we will also use “task” in a more specific context (i.e. representing a specific step or activity within an overall data mining process such as CRISP-DM).

Confronted with the fact that the use of data mining has become a very specialized field, at first hand it may not seem obvious how one can effectively profit from the use of DM technology. A steep learning curve may be required up front, the process is littered with many grueling and technical decisions, and there are no guarantees that a successful effort shall satisfy the intended business objectives.

2.2 Knowledge Representation and Reasoning Challenges

While the previous section has focused on the intrinsic challenges associated with the field of data mining, this section shall briefly examine the specific challenges associated with applying both the CBR and ontology knowledge representation formalisms for a given application domain.

2.2.1 Case-Based Reasoning

Although we shall address the fundamentals of CBR systems in Section 3 and subsequently particular CBR design considerations in Section 4, (i.e. case vocabulary, indexes and similarity measures, adaptation strategies, seed case and case maintenance issues), Figure 3 illustrates some of the key challenges associated with implementing and exploiting the CBR paradigm. The figure clearly illustrates some of the important (and strongly inter-dependent) design decisions that must be considered prior to implementing a CBR application (i.e. similarity measures, indexing mechanisms, case library representations and possible adaptation methods).

From a designer's perspective, the realization of a CBR system can be viewed as appropriately making use of design choices available from five different toolboxes (i.e. *Case Representations*, *Similarity Measures Toolbox*, *Indexing Methods Toolbox*, *Adaptation Methods Toolbox* and *Case Library Structures*). For instance, the designer must not only effectively capture the problem domain as a set of representative problem (indexes) and solution features with associated data types (i.e. string, integer, float, etc.), she must also select appropriate local and global similarity measures (similarity measures are covered in detail in Section 4.3). To complicate matters, a designer may

have to investigate if the use of surface similarity measures⁴ is adequate for the application domain, or if more complicated structural similarity measures are required [60]. It is worth noting that such a decision may have to be postponed until a working CBR system prototype has been realized and validated.

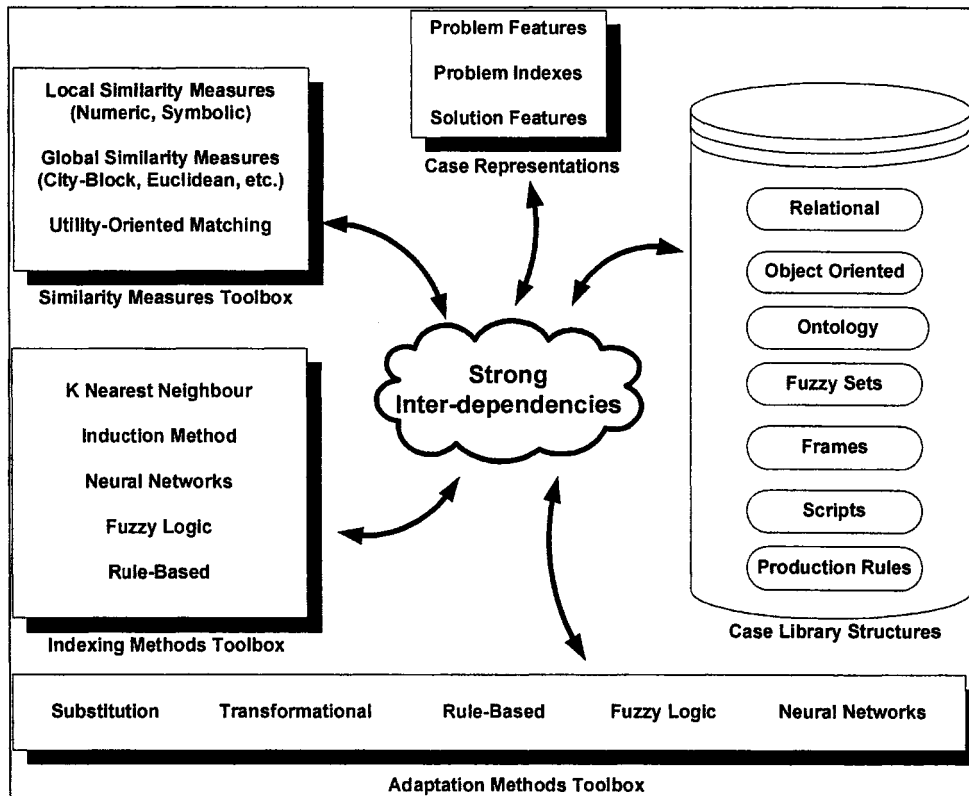


Figure 3 Some Challenges Associated With Exploiting the CBR Paradigm

Consequently, the chosen case representation and similarity measure details can have an impact on the possible low level case library representation structure which can be used (i.e. object oriented, frame-based, fuzzy set, etc.). Furthermore, depending on the preceding design choices, an appropriate indexing method must be considered for case retrieval (i.e. K nearest neighbor, induction method using a decision tree, fuzzy logic, etc.). Last, but not least, depending on the complexity of the application domain, the

⁴ See Section 3.4.4 for more details.

choice of an appropriate adaptation method may be required (i.e. rule-based, neural network based, etc.) [76].

2.2.1.1 Limited Availability of CBR Frameworks

A preliminary evaluation of CBR literature and web resources ([3], [112]) has confirmed that, though many academic CBR frameworks were developed over the past decade, the recent availability of these frameworks for research purposes has proven very difficult. In 1997, Watson evaluated a series of CBR frameworks (i.e. CBR-Express, Remind, Recall, Kate-CBR, CBR-Works, CBR*Tools, CREEK, etc.), unfortunately most of these are no longer available either as prototypes or commercial products [113]. In fact, though we shall review some existing commercial and academic CBR frameworks in Section 3.4.7, the difficulty of acquiring an adequate CBR framework for our research was the principal motivating factor behind why we chose to implement our own minimal CBR framework (see Section 4.3.2).

2.2.1.2 The Need for a KI-CBR Framework

Knowledge-Based Systems achieve their reasoning power through the explicit representation and use of different kinds of domain specific knowledge. Although the primary knowledge source for traditional CBR systems is often represented via a set previous cases or experiences, the effective application of CBR systems within complex application areas (i.e. medical diagnosis, data mining assistance, etc.) often requires a supplementary knowledge source in order to successfully carry out the retrieval or revision phases of the CBR cycle and achieve its intended purpose (i.e. diagnosis, prediction, planning, etc.) [28]. Though this KI-CBR approach can be effective for solving complex problems, it also gives rise to new challenges: (1) the introduction of a second knowledge source implies the need for additional knowledge acquisition efforts; (2) an effective “bridging” or integration strategy may be required in order to efficiently exploit two disparate knowledge sources. (See Section 4.4 for details on how these challenges and many other related issues were addressed).

2.2.2 OWL DL Ontologies

Though OWL DL ontologies can be very effective as a knowledge representation and reasoning formalism, the exploitation of formal OWL DL ontologies poses its own set of unique challenges. The following is a concise, non-exhaustive list of some of the current limitations associated with the use of OWL DL ontologies:

- **Expressivity vs. Decidability Trade-off** - The OWL DL language was carefully specified in order to provide limited “expressivity” using the *SHOIN(D)* DL family (i.e. a decidable subset of FOL that exhibits worst-case non-deterministic exponential time complexity) to ensure decidability [43].
- **Default Values and Knowledge Elicitation** - OWL does not currently support the use of inheritable default values (as is often employed with the object-oriented paradigm and can facilitate the knowledge acquisition effort) [42].
- **Rules and Procedural Knowledge** - OWL ontologies currently only provide limited procedural knowledge support via the proposed SWRL [103] standard (i.e. rules are restricted to Horn-like clauses, rules cannot be expressed as properties, etc.).
- **Integration with Rule-Based Reasoners** - Though OWL is currently reasonably well integrated with DL reasoners (i.e. Racer [83], Pellet [77]), the same cannot be said about the integration of rule-based reasoners (i.e. CLIPS [24], JESS [50]).
- **Ontology Evolution and Maintenance** - Like any other knowledge representation formalism, knowledge engineers using OWL must contend with the particular challenges associated with eliciting and maintaining domain knowledge in a clear, consistent and (ideally) complete form.

2.3 Data Mining Without Assistance – A Simple Example

In order to build a case for intelligent data mining assistance (and to get a better appreciation for the difficulties involved with the practice of data mining), the following is a brief example demonstrating the intricate and often difficult choices that a novice miner is faced with during the course of a typical data mining task. For all intents and purposes, we shall assume that the user has already selected a problem data set of interest and that the intended business and DM objectives are reasonably well defined (i.e. a regression problem where the target label has been identified). To fully appreciate the difficulties and associated detailed DM knowledge required for carrying out a “correct” DM activity without assistance would require one to exhaustively analyze and understand all the activities specified within a DM process such as CRISP-DM. Nonetheless, we hope the following brief example will provide convincing evidence for novice data miners to endorse the use of intelligent data mining assistants.

Since a problem dataset of interest is rarely ever in a perfectly suitable format for immediate model generation (i.e. presence of missing, outlier, incomplete, invalid and duplicate values), Figure 4 demonstrates some of the feature (attribute) transformation filters available with the Weka DM toolkit [115]. We have only shown a small portion of the available filters (i.e. *Discretize*, *Normalize*, *NumericToBinary*, etc.), nonetheless one can easily imagine how such a variety of possible options (early-on within a DM process) can be quite overwhelming for a novice data miner.

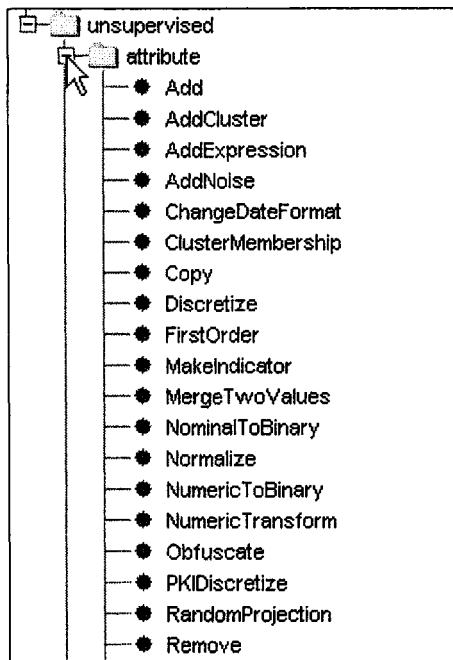


Figure 4 Selecting a Feature Transformation Filter Using Weka (source [115])

The decision required by a non-specialist data miner for choosing an appropriate algorithm is by no means any easier. For example, though Figure 5 only displays the “trees” family of algorithms, the Weka DM toolkit currently makes available over 70 data mining algorithms (grouped into 6 distinct families), each with its respective merits depending on the particular needs of a DM application. The sheer detailed DM knowledge required by an expert data miner to fully exploit this particular DM toolkit is a tremendous challenge, one can only imagine the difficulties that await a novice data miner or decision maker.

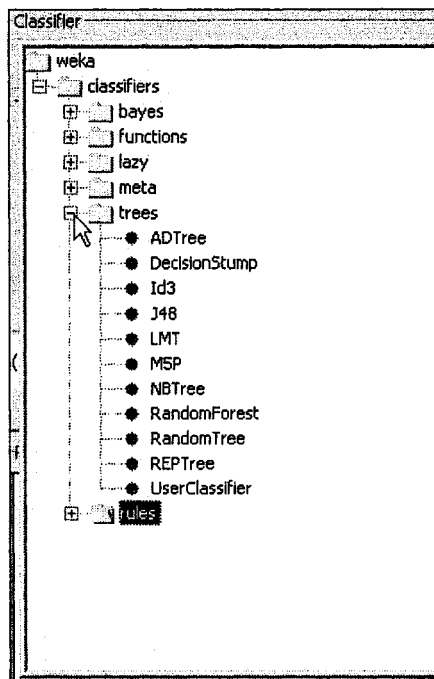


Figure 5 Selecting a Machine Learning Algorithm Using Weka (source [115])

To complicate matters further for non-specialist miners, each DM algorithm also possesses a series of settings or parameters that may require adjustment during the training process in order to obtain better results (i.e. error rate). In reality, the appropriate use of the parameters is very much specific to the chosen algorithm. Users are frequently referred to journal articles in order to understand the finer details of the algorithm in order to make a decision of whether to apply the parameter defaults or make a specific parameter selection. Such a recommendation may be a moot point for novice data miners and Figure 6 serves to further demonstrate the level of technical complexity involved with the practice of data mining. More specifically, Figure 6 illustrates the 14 possible parameter settings that comprise the Weka implementation of the Sequential Minimal Optimization (SMO) algorithm.

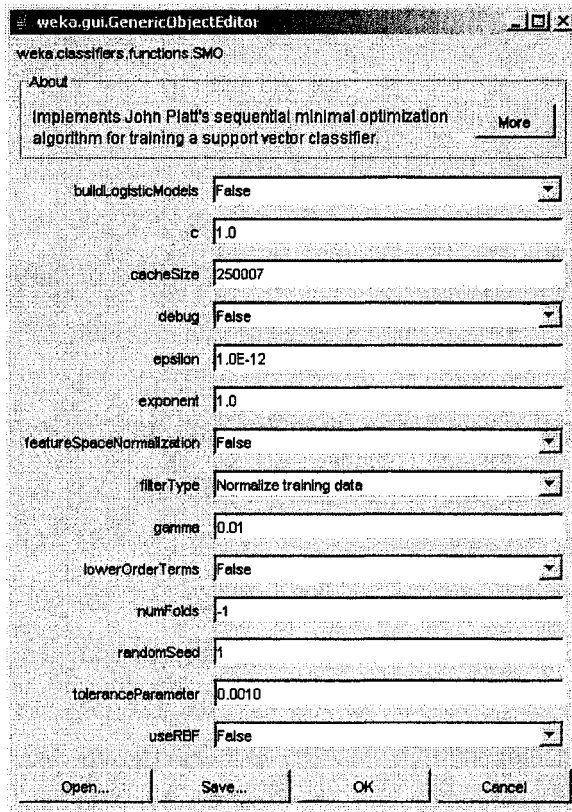


Figure 6 Setting Training Parameters Using Weka (source [115])

Last, but not least, assuming that the novice miner is able to survive the above data understanding, data preparation and data modeling steps, she must still possess adequate knowledge for evaluating the “quality” of the obtained generalization (model). For example, Figure 7 displays a typical summary after the generation of a model using the SMO algorithm. One observes from the figure that the results consist of a Kappa statistic, 4 different residue error measures (i.e. mean absolute, root mean squared, relative absolute and root relative squared), Precision, Recall, an F-measure and a confusion matrix. Which of these evaluation parameters is the most appropriate for assessing the quality or performance of the overall DM activity? This is exactly the kind of situation (amongst many others) where an effective data mining assistant can best serve the interests of a novice data miner.

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      144           96    %
Incorrectly Classified Instances     6            4    %
Kappa statistic                     0.94
Mean absolute error                 0.0287
Root mean squared error             0.1424
Relative absolute error             6.456 %
Root relative squared error         30.2139 %
Total Number of Instances           150

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  Precision  Recall  F-Measure  Class
  1       0       1          1       1          Iris-setosa
  0.92    0.02    0.958     0.92    0.939     Iris-versicolor
  0.96    0.04    0.923     0.96    0.941     Iris-virginica

=== Confusion Matrix ===

  a  b  c  <-- classified as
50  0  0 | a = Iris-setosa
 0 46  4 | b = Iris-versicolor
 0  2 48 | c = Iris-virginica

```

Figure 7 Examining the Model Evaluation Parameters Using Weka (source [115])

Having briefly witnessed some of the difficulties that plague non-specialist data miners, Section 4 shall address the key implementation details that were carried out for implementing an intelligent DM assistant capable of managing some of the aforementioned DM challenges.

2.4 A Formal Research Objectives Statement

As previously mentioned in Section 1.3, we have proposed a theoretical, conceptual and technological framework for the development of an intelligent DM assistant. Furthermore, we have attempted to make use of various knowledge representation and reasoning formalisms (cases, concepts, rules) and associated reasoning paradigms (i.e. case-based, ontology-based and rule-based reasoning) in order to achieve the DM assistance challenges mentioned in this section (Section 2.1 and 2.2). More specifically,

we have tested if the combined use of a CBR and formal DL ontology can support the reuse of DM knowledge for decision support purposes and aiding non-specialist miners.

2.4.1 The UQTR Decision Support Department - A Real Case Study

During the course of our research, we have been fortunate enough to have the opportunity to work closely with a decision support group at UQTR (*la direction des affaires départementales*). Particularly, an analyst within the group is responsible for producing predictive models (i.e. correlation, linear regression, classification) using large quantities of student data (i.e. student registration data, academic profiles, student surveys, etc.). The department had manifested an interest in potentially using data mining tools and techniques for accomplishing their modeling goals. As a result, we have had the opportunity to test our intelligent DM assistant using “real world” data. Some of the DM results obtained from this case study are elaborated upon in Section 5.

Having been introduced to some of the key challenges associated with providing intelligent data mining assistance for novice data miners, the following chapter provides a detailed overview or state-of-the-art of the key components that were considered during the realization of our proposed intelligent DM assistant.

CHAPTER 3

State of the Art

The following chapter provides basic definitions, concepts and a brief state of the art for each of the key elements (i.e. CRISP-DM, meta-learning, CBR, Ontologies, etc.) that were eventually integrated within our intelligent data mining assistant implementation. First, since these play a foundational role within our project, we shall address DSS technology and the CRISP-DM data mining process. Subsequently, we briefly present a state of the art for meta-learning, CBR and formal DL ontologies. Last, we shall discuss some of the finer points of KI-CBR and rule-based expert systems. With respect to our discussions on CBR and ontologies it is important to note that we provide a slightly more detailed discussion on the former. This may be explained by the fact that the CBR paradigm has played a more prominent role in implementing our DM assistant.

3.1 Decision Support Systems

Although there is no universally accepted definition for a DSS, the following multi-faceted definition from Turban *et al.* provides a clear direction [107]:

“A DSS is an approach (or methodology) for supporting decision-making. It uses an interactive, flexible, adaptable computer-based information system especially developed for supporting the solution to a specific non-structured management problem. It uses data, provides an easy user interface, and can incorporate the decision-maker’s own insights.”

Overall we tend to agree with this definition of a DSS, however we prefer to extend or generalize the intended user base for DSS beyond that of “management problems” (management decision-makers) to supporting the resolution of all forms of non-structured problems (i.e. data mining) and associated decision makers within an organization (i.e. engineering, finance, medical, etc.).

3.1.1 The Anatomy of a DSS

As illustrated in Figure 8, a DSS mainly consists of the following 4 components:

- **Data Management Subsystem** – A repository to store and manage relevant application domain data.
- **Model Management Subsystem** - A repository to store quantitative models (i.e. financial, statistical, etc.) that provide analytical capabilities.
- **Knowledge Management Subsystem** – A component that provides intelligence to augment the decision maker's own. This subsystem can support any of the other subsystems.
- **User Interface Subsystem** – an interface used by the user (decision-maker) to command the DSS.

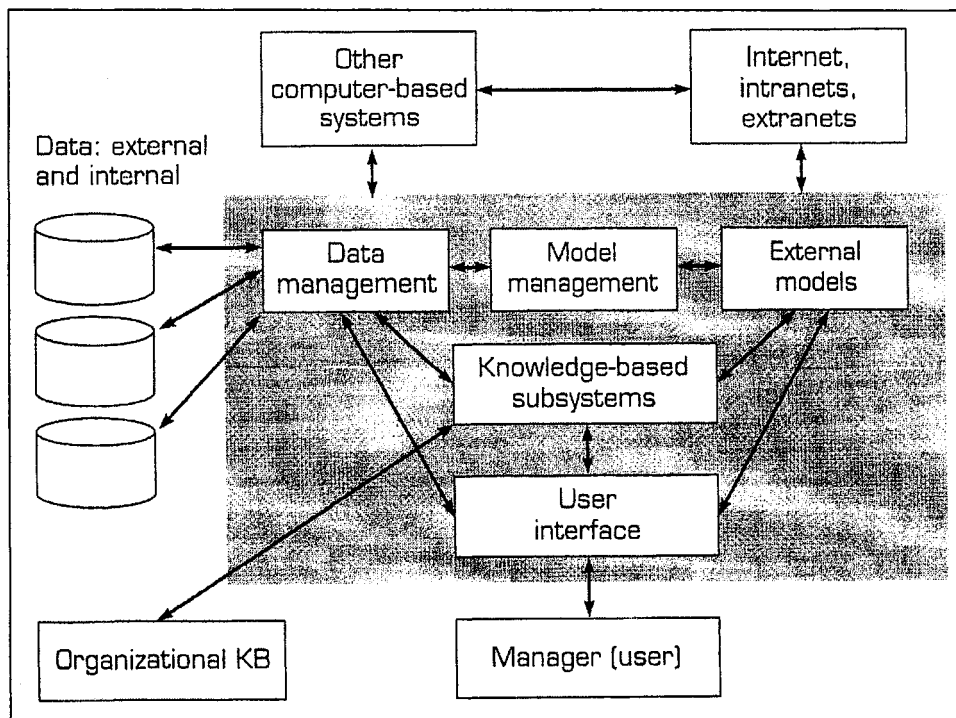


Figure 8 - A Schematic View of a DSS (source [107])

3.1.2 Applications Areas for DSS

Though early developments and application of DSS were mainly focused in the area of enterprise management, due to its proven effectiveness at supporting the general decision-making process, over the past several decades DSS have found widespread usage in a number of domains [62]:

- **Environment** – During the early 1980's DSS have been deployed to support decision-making in the area of environmental impact assessment (i.e. watershed levels and quality).
- **Agriculture** – DSS are used in order to globally improve agricultural productions processes (i.e. managing irrigation schedules in harsh climate environments).
- **Aviation** – Systems are used, both commercially and from a military perspective, at tasks ranging from flight scheduling, flight-path planning, air-traffic control to specific applications such as aircraft landing assistance.
- **Manufacturing** – DSS are used for a plethora of activities ranging from supply chain management, customer relationship management to finance and investment.
- **Medical** – Medical expert systems have been used for decades in order to support decision-making in domains ranging from disease diagnosis, toxicology, and cancer research to surgery and emergency situation management [31].

3.1.3 Knowledge Management Systems

Previous knowledge and expertise (i.e. know-how) within an organization can often be re-used to support current decision-making needs. It does not make much sense to continually reinvent the wheel each time a new problem or situation is encountered. Rather the knowledge accumulated over time can be used to solve identical or similar problems (i.e. a corporate memory). When considering the management of knowledge, there are several important issues to address: how to classify and store knowledge,

how to maintain and ensure the quality of knowledge, how to retrieve, find and effectively leverage its use [107].

A Knowledge Management System (KMS) and its associated technologies (i.e. DBMS, Data Mining, etc.) are designed to deal with some of the aforementioned issues. Since most of the knowledge within an organization is non-documented (i.e. tacit knowledge) and a significant amount of tacit knowledge is lost every time people retire or leave an organization, enterprises have a strong vested interest in putting a KMS in place in order to counter the potentially adverse affects of a “knowledge-drain” effect. When comparing a KMS to a KBS, though both systems can address some or all of the above mentioned issues (i.e. storage, maintenance and retrieval of knowledge), a KMS is most commonly associated as being an “umbrella” system which can incorporate one or more KBS. As previously discussed in Section 3.1.1, a DSS architecture is typically composed of a KMS component for supporting and improving the overall decision-support effort. It is worth mentioning that overlap and cross-functionality do tend to exist between a DSS, KBS and KMS.

3.1.4 Data Mining Assistants

The following brief survey or stat-of-the-art on DM assistants shall primarily focus on proposed solutions based on the use of meta-learning (i.e. machine learning at the meta-level), the CBR-paradigm or formal ontologies. While some of the proposed DM assistants have focused on the data preparation phase (i.e. feature selection), as previously mentioned in Section 2.1.4, the majority of the efforts have concentrated on providing users with model selection support. First, Kalousis *et al.* [52] have developed a CBR-based assistant named *NOEMON* for selecting an appropriate classification or regression DM model. Second, the successful *MetaL* [65] project provides an on-line advisory system, based on an instance-based machine learning algorithm (i.e. K-Nearest Neighbor), that uses data and performance characteristics to characterize the data mining problem and offers a ranking of suitable DM algorithms. Third, Lindner and Studer [59] have proposed a model selection assistant named *AST* (Algorithm Selection Tool) which characterizes a DM problem based on application restrictions, the problem dataset and user experience. Fourth, Bartlmae [5] has provided a framework, based on

the CRISP-DM methodology, the CBR and Experience Factory paradigms, for capturing coarse-grained knowledge (i.e. lessons, guidelines, documentation, etc.) during the mining process.

Several previous research efforts have demonstrated the effectiveness of using formal ontologies for supporting the knowledge discovery process. First, Bernstein *et al.* [10] have proposed an intelligent data mining assistant based on the use of an ontology. Their ontology contains constraints and performance knowledge that is eventually searched for in order to find a ranking of possible satisfactory DM processes. Second, Phillips and Buchanan [79] have used ontologies to guide the feature selection step of the knowledge discovery process. Third, Bauer and Baldes [6] have used an ontology-based interface to aid non-expert users of machine learning better understand and influence an ML system from a semantic perspective. Fourth, Canataro and Camito [16] have demonstrated the use of a DM ontology to simplify the development of distributed knowledge discovery applications in the area of grid computing. Last, Suyama *et al.* [100] have developed a platform named CAMLET, for the automatic composition of inductive learning applications using an ontology.

3.2 The CRISP-DM Process

Although other proprietary DM processes have been defined over the past decade (SEMMA [90], Affinium [109]), CRISP-DM [18], having become well established within various sectors of industry and virtually the *de facto* DM process, provides a form of process knowledge that can foster better chances of an overall DM project's success (though it cannot guarantee a successful outcome). As illustrated in Figure 9, the data mining process is organized into six principal phases; each phase consists of several second-level generic tasks. Subsequently, each task can be further subdivided into activities with associated inputs and outputs. The arrows indicate the dependencies between the phases.

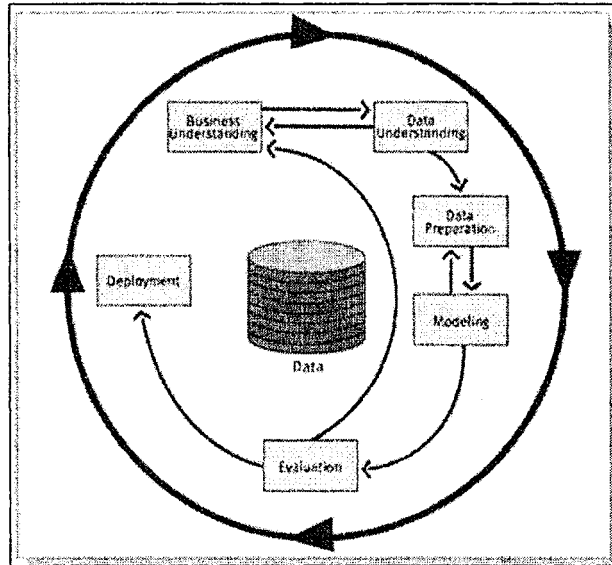


Figure 9 The CRISP-DM Process Model Life-Cycle (source [25])

The following is a concise description of each of the six phases of CRISP-DM [25]:

- **Business understanding** - This phase focuses on understanding the project objectives and requirements from a business perspective, then converting this knowledge into a data mining problem definition.
- **Data understanding** - The purpose of this phase is to initiate data collection and perform a high-level analysis in order to get familiar with the data (i.e. identify data quality problems, possible required transformations, data semantics such as missing and invalid values, etc.).
- **Data preparation** - The data preparation phase covers the activities to construct the final dataset that will be applied during the modeling phase. Common tasks include table, record and attribute selection as well as transformation and cleaning of data for modeling tools.

- **Modeling** - In this phase various modeling techniques are applied (by modifying parameters and carrying out appropriate data preparations) in order to acquire the best model performance seek the best model performance. Model performance is typically assessed using one or more measures (i.e. P-value, residue errors, error rate, Kappa value, etc.).
- **Process Evaluation** – In this phase, the entire DM process is reviewed and evaluated to assess if it has met the previously established business and DM objectives (*Business Understanding* phase).
- **Deployment** – The purpose of this final phase is to define how the generated model or gained knowledge will be integrated within the operating business environment (i.e. use the model with a problem dataset and produce a report with results, integrate the model within an application such as a web-site for repeated use, etc.).

Figure 10 highlights the basic structure and key activities that should be carried out during the data preparation phase of the CRISP-DM process. The data preparation phase consists of 5 main activities (i.e. *Select Data, Clean Data, Construct Data, Integrate Data and Format Data*). As illustrated in the figure, if carried out correctly, these main activities produce approximately 8 associated outputs or deliverables (i.e. *Derived Attributes, Generated Records, etc.*).

As previously mentioned, though the CRISP-DM process model is effective at thoroughly specifying the required tasks and activities from a general standpoint, it does not provide specific details or “know-how” for carrying out the particular steps of the process. Evidently, this is not a shortcoming of the CRISP-DM process in itself, but rather a fact that, in order to effectively support a non-specialist data miner, the process should be leveraged with a complementary knowledge source (see Section 4.4 for more details).

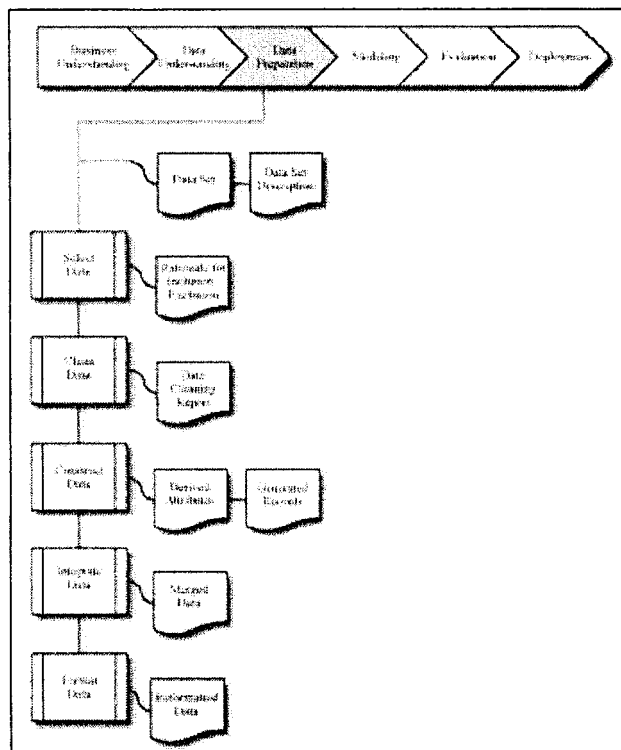


Figure 10 The CRISP-DM Data Preparation Phase Flowchart (source [25])

3.3 Meta-Learning

Although many researchers hold different views of what meta-learning is, the fundamental problem lies with the fact that there exists a plethora of machine learning algorithms for DM (with numerous parameter settings), but very limited support for the selection of an appropriate (ideally optimal) model for a given DM task. The situation is further complicated by the practical impact of Wolpert's "No Free Lunch" theorem, which substantiates that no given DM algorithm significantly outperforms all others for all conceivable DM applications [38]. While base learning (i.e. data mining) is focused on accumulating experience (i.e. model) on a specific learning task (i.e. medical diagnosis, fraud detection, credit rating approval), meta-learning is concerned with accumulating experience over the performance of multiple applications (possibly across multiple application domains) of a learning system. From a practical stance, meta-learning is concerned with helping to solve important problems for the effective application of machine learning and data mining tools. As previously mentioned in

Section 2.3, such problems can range from inadequate data preparation and model selection to the inappropriate application of training parameters and model evaluation methods. For instance, though Leite and Brazdil [57] have carried out meta-learning research explicitly to support the data preparation phase, the following discussion on meta-learning shall primarily focus on model selection support. Our survey of the research literature on meta-learning can be summarized by the following key objectives or research areas⁵:

- **Model Selection Assistance:** A system that provides support for the selection of an appropriate algorithm for a given DM task.
- **Learning-to-Learn:** A system capable of learning from experience. For example, if a system behaves poorly for a given task, it should be able to learn and improve its performance the next time the same task is attempted.
- **Inductive Transfer:** A system capable of reusing learned invariant properties across many related application or domains.

3.3.1 Dataset Characterization

Although not a first-order objective in itself, high-quality data characteristics can provide information for differentiating the performance of a set of learning strategies. Since data characteristic measures (i.e. meta-features) play such a pivotal role in most meta-learning systems, the following presents some of the key techniques used.

3.3.1.1 General, Statistical and Information-Theoretic Measures

Three categories of data characterization measures that have gained widespread acceptance in the past decade are general, statistical and information theoretic measures ([40], [19]). General measures mainly consist of simple dataset features such as the number of examples, number of attributes, number of classes, ratio of missing values, etc. Common statistical measures mainly comprise of correlation measures

⁵ Meta-learning also encompasses other areas such as ensemble learning, dynamic bias selection, implicit culture models using multi-agent systems and auto-adaptive algorithms. Nonetheless, due to space limitations, our survey was constrained to the above key areas as these pertain to our research.

between features and the target concept, *kurtosis* and *skewness*, while information-theoretic measures typically consist of *average class entropy*, class-conditional entropy and *maximum mutual information*. Interestingly, these types of measures were used in the aforementioned DM assistant projects such as NOEMON [52], AST [59] and the MetaL project [65]. Section 4.3.4 shall elaborate in more detail on the particular measures chosen for implementing our intelligent DM assistant.

3.3.1.2 Model-Based Characterization

In addition to statistical measures, model-based characterization is another form of measure that exploits properties from an induced model. For example, having generated a decision tree from a given dataset, useful characteristic measures such as nodes per feature, maximum tree depth, tree shape and tree imbalance can be obtained and further exploited [78].

3.3.1.3 Landmarking

Another method of characterization known as landmarking is based on the principle of exploiting information obtained from the performance of a set of simple learners. The results (i.e. error rate) obtained from training a dataset using a set of simple learners (i.e. landmarkers) can subsequently serve as a characteristic measure to identify areas where similar datasets are most likely to perform well for a given learning strategy [8].

3.3.2 Model Selection Assistance

An important and practical objective of meta-learning is the construction of a mechanism that maps an input space composed of datasets (i.e. application domains) to an output space composed of learning strategies (i.e. predictive models). Performance criteria such as accuracy, storage space, and running time can be used as a target concept. Two of the most popular methods for mapping datasets to predictive models are described below.

3.3.2.1 Learning at the Meta-Level

As illustrated in Figure 11, learning at the meta-level (i.e. similar to a base-level machine learning task) consists of inducing a “meta-model” from the use of meta-dataset. The meta-dataset consists of labeled examples where the input features are data characteristics and the target concept corresponds to the model value with the best performance on the original problem dataset (i.e. a set of (f_i, m_i) pairs, where f_i is a vector of data characteristics and m_i is a corresponding model). Interpreting the meta-learning problem as a standard machine learning task, the main objective is to induce a hypothesis ($E(f_i) \sim m_i$) or model using a learning strategy (i.e. classification algorithm) that can map dataset characteristics to predictive models. Though a host of machine learning algorithms can be applied to this ML problem, Berrer *et al.* have substantiated the effectiveness of using an instance-based, feature-weighted learning approach such as K-nearest neighbor to obtain a model with the best average performance [11].

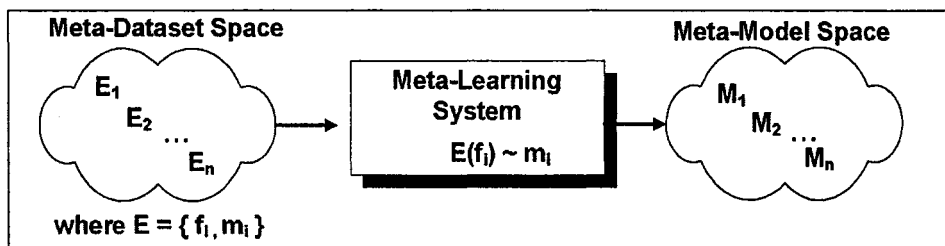


Figure 11 Learning at the Meta-Level

3.3.2.2 Model Ranking Methods

Instead of mapping a dataset to a single predictive model, one may also produce a ranking over a set of different models. As illustrated in Figure 12, a ranking can provide alternative solutions to users who may wish to incorporate their own expertise into the decision-making process. Various model ranking approaches have been suggested such as an IBL-based approach by Brazdil and Soares [14] and a ranker based on the Normalized Mean Squared Error (NMSE) measure by Gama *et al.* [33].

Algorithm	Recommended Rank	Target Rank	Accuracy %	Time s
Boosted C5.0	1	1	95.3	77
IB1	2	2	93.6	163
Linear Discriminant	3	8	70.2	2
Ltree	4	4	86.9	397
C5.0 (rules)	5	3	88.8	222
C5.0 (tree)	6	5	87.9	8
Naive Bayes	6	9	64.4	10
RIPPER	8	6	86.2	1249
Radial-Basis Function Network	9	10	43.9	4946
MultiLayer Perceptron	10	7	79.8	3998

Figure 12 Model Ranking Example for a given DM Problem (source [111])

3.3.3 The Learning-to-Learn Paradigm

Machine Learning should not be viewed as an isolated task that starts from scratch on every new problem. Rather, the learning-to-learn paradigm takes the stance that the learning mechanism should strive to accumulate experience and ideally perform increasingly better over time. As previously mentioned, one of the key objectives of meta-learning should be the realization of a system that is capable of learning from experience to eliminate the repetition of the same mistakes or having to continuously restart the learning process from scratch. The goal of the Learning-to-Learn paradigm is not to map datasets to predictive models (as in learning at the meta-level), but rather to continuously integrate new meta-knowledge over a range of learning tasks and improve the performance of such tasks over time. For instance, a CBR system, by its capacity to accumulate experience over time in the form of cases, is a simple, yet effective, realization of an application possessing learning-to-learn characteristics.

3.3.4 Inductive Transfer

A closely related principle to the Learning-to-Learn paradigm is inductive transfer. Inductive transfer seeks to leverage the use of supplementary knowledge sources from other application domains (i.e. learned invariant properties) in order to improve the

performance of a task within a given domain. According to Thrun, inductive transfer is inextricably tied to learning-to-learn as it provides a necessary condition for effectively realizing the Learning-to-Learn paradigm [105]. For example, Caruana *et al.* [17] explain that training a single neural network in parallel with related domains (i.e. physiotherapy, occupational therapy) induces information that accumulates in the training signals. As such, a new or related domain (i.e. osteotherapy) can immediately benefit from such past experience.

3.4 Case-Based Reasoning

3.4.1 Definition

The field of CBR arose out of research in cognitive science and early contributions in this area were from Roger Schank and his colleagues at Yale University [92]. As we shall see shortly, this methodology or paradigm was shown to be useful in a wide range of AI-related applications. Unlike most problem solving methodologies in AI, CBR is memory-based and thus reflects human use of remembered problems and solutions as a basis or starting point for resolving a new problem. Essentially, the problem-solving approach used by a CBR system consists of the following phases:

- **Problem Characterization** - Obtain a suitable problem description for the current problem of interest.
- **Retrieval** - Measure the similarity of the current problem with previously resolved problems residing within a case base and retrieve one or more similar cases.
- **Reuse** - Evaluate and attempt to reuse the solution of one of the retrieved cases.
- **Revision** - If necessary adapt the chosen basis case (due to problem description discrepancies) to resolve the problem in question.

- **Retain** - If the solution is successful, retain the resolved problem or case for future reference.

Moreover, Figure 13 clearly illustrates the above-mentioned phases in the form of life-cycle that (ideally) continuously processes new problems and eventually retains them for future use. Hence, a CBR system acquires “experience” over time in the form of cases which can later be used to solve similar (or related) problems.

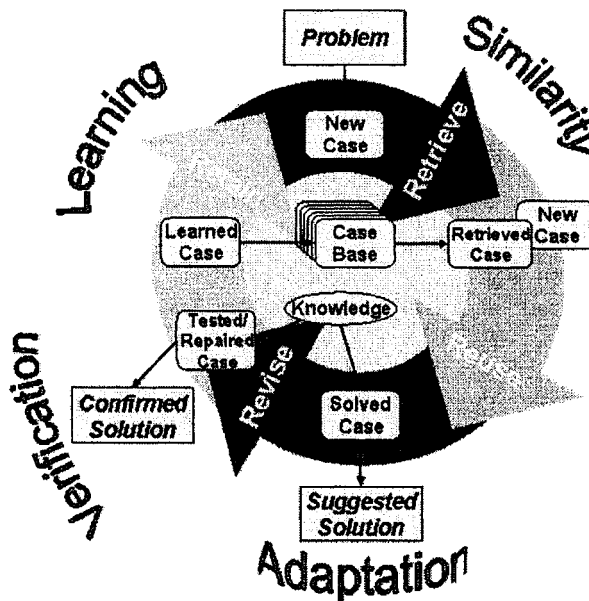


Figure 13 The Case-Based Reasoning Cycle (source [86])

As we shall see shortly, the *retrieval* phase is commonly implemented using various similarity measures, while decisions related to *reuse* or *revision* of a case may be assisted by a supplementary knowledge source or intelligent component (i.e. machine learning or knowledge-based system). The *retain* phase is typically simple (involving merely saving the case), however strategies have been recommended for further validating the nature of a solution or case prior to retention. It is worth mentioning (and shall be demonstrated more thoroughly in Section 4) that the difficulties with implementing an effective CBR system particularly lie within the sophistication and complexity requirements for implementing both the retrieval and revision phases.

3.4.2 Some Advantages of Using CBR

In this section we summarize some of the advantages of using a CBR system [76]:

- **Reduce Knowledge Acquisition** – By eliminating the need to extract a formal model or set of rules from previous experiences (as is the case for rule-based or model-based systems), CBR knowledge is usually less formal and structured. Hence, this can be more convenient and intuitive for users to gather and store knowledge in the form of cases.
- **Avoid the Repetition of Mistakes** – Systems that record successes and failures may be used to predict or avoid failures for future problems.
- **Reasoning in Domains not Fully Understood** – In application areas where domain knowledge is not fully known, formalized or quantified, a CBR system can still be effectively deployed.
- **Learning Over Time** – As cases are added to a CBR system⁶, it is able to reason over a wider variety of situations and with a higher degree of precision.
- **Reasoning in a Domain with Limited Knowledge** – In a problem domain where only a few cases are available, a CBR can start with these few known cases and build knowledge incrementally as cases are added.
- **Problem Solving Time Efficiency** – In application areas where the resolution of a case requires a significant amount of time and effort, the alternative offered by a CBR for reusing or adapting a past basis solution can be very attractive.

⁶ Evidently, an important assumption is that periodic maintenance is carried out to eliminate harmful cases and ensure good case coverage (see Section 3.4.6 for details).

- **Applicable to a Broad Range of Domains** – Since an unlimited number of ways exist for representing, retrieving and adapting cases, CBR can and have been applied in extremely diverse application areas (see next Section 3.4.3).

3.4.3 Common Applications of CBR

As indicated by Figure 14, CBR applications are broadly classified into two main problem types: classification and synthesis tasks. A classification task can be recognized by a need to match, as closely as possible, an object against others in a library from which a solution can be inferred. On the other hand, a synthesis task attempts to create a new solution by combining parts of previous solutions. Synthesis tasks are intrinsically more complex because of the constraints or dependencies between the elements used during the synthesis process. Essentially, a classification task requires recognition of similar features, whereas the synthesis task requires placing the correct features in the correct order.

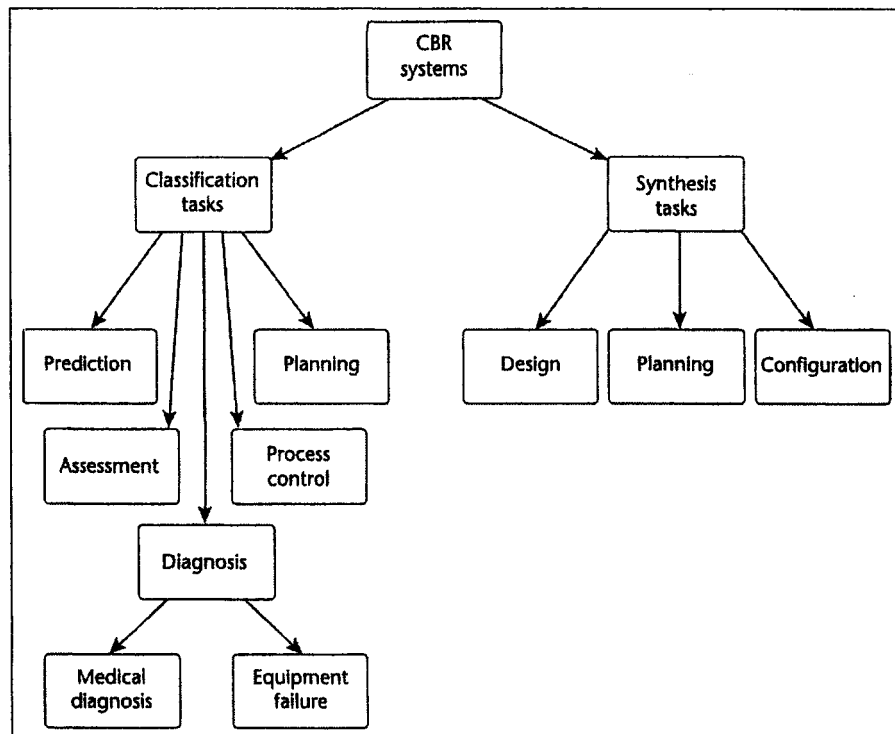


Figure 14 A Classification Hierarchy of CBR Applications (source [113])

Classification tasks are further organized into the following sub-categories:

- **Diagnosis** – Medical diagnosis or equipment failure diagnosis.
- **Prediction** - The forecasting of equipment failure or stock market performance.
- **Assessment** – Risk analysis for banking or insurance.
- **Process control** – The control of manufacturing equipment.
- **Planning** – The reuse of travel plans, work schedules or project estimates.

Similarly, synthesis tasks are sub-categorized into the following areas:

- **Design** – Creation of new artefacts by adapting elements of previous artefacts.
- **Planning** – The creation of new plans from an ensemble of sub-plans.
- **Configuration** – The creation of new schedules from previous ones.

If the above hierarchical classification of CBR systems is any indication, over the past decade, CBR systems have been implemented and successfully applied in far too many areas to mention in this report. For instance, Watson [113] lists more than 130 major companies in all sectors of industry (i.e. hardware and software technology, finance and insurance, telecommunications, manufacturing, transportation, retail, etc.) which make use of CBR on a daily basis.

3.4.4 Retrieval and Similarity Assessment Methods

As previously mentioned, an important step in the CBR cycle is the retrieval of previous cases that can be used to resolve the target problem. Although Lopez de Mantaras *et al.* [60] have surveyed many other approaches for retrieval in CBR systems (i.e. adaptation-guided, diversity conscious, compromise-driven, etc.) and Liao *et al.* [58] offer an exhaustive treatment of possible similarity-based retrieval techniques (i.e. using crisp sets, fuzzy sets, hybrid measures, etc.), the following discussion shall focus on commonly used similarity-based retrieval techniques.

3.4.4.1 Surface Similarity Approaches

Informally, the surface similarity between two cases is a numerical measure of the degree to which the two cases are alike. The similarity is typically computed using a mathematical function that computes a real number between [0,1] (where 0 implies no

similarity and 1 represents exactness). The surface features of a case (also known as the indexing vocabulary) are those representative features that are provided as part of the case description and are usually represented as attribute-value pairs. In order to implement a similarity measure, transformations are often applied to convert the indexing features (i.e. binary, categorical, unbounded numerical, etc.) to satisfy the above-mentioned numerical constraint (i.e. [0,1]). The following are some of the commonly used surface similarity-based retrieval approaches:

- **Nearest-Neighbour Retrieval** - a common surface similarity-based retrieval approach known as k nearest neighbour (i.e. k-NN) retrieves the k most similar cases with respect to the target problem. First, each case is represented as a simple feature vector of attribute values. For each case comparison, local similarity measures are defined for each attribute and an overall global similarity calculation is then computed (as a weighted average of the local similarity measures) [114].
- **Tree-Based Retrieval** – is a method using a binary tree (i.e. k-d tree) that organizes the case-base into groups or clusters according to a given similarity measure [116].

3.4.4.2 Structural Similarity Approaches

Though computationally more expensive because it relies on the extensive use of domain knowledge, retrieval based on structural similarity has the advantage that more relevant cases may be retrieved. As previously mentioned in Section 2.2.1, the nature and complexity of the problem domain may dictate the use of a structural similarity measure (or a combination of surface and structural similarity) to obtain adequate retrieval quality. Borner [13] defines structural similarity as the most specific graph structure that the target problem has in common with a stored library case and the associated background knowledge (i.e. transformation rules) required to assess the similarity. The following are a few of the structural similarity-based retrieval methods that have been proposed:

- **Object-Oriented Similarity** – this method represents cases as sets of objects. These objects or “cases” belong to classes that are organized into a class hierarchy. Objects that are closer to each other in the hierarchy are likely to be more similar. For example, Aamodt *et al.* [1] have proposed Creek, a CBR framework which uses an object-oriented system to capture cases and domain knowledge. Subsequently, the case-base can be viewed as a multi-relational semantic network.
- **Generalized Cases** – a method that represents relations between attributes using generalized cases. A generalized case covers a region of the problem-solution space rather than just a single problem. Amongst others, Bergman *et al.* [60] have proposed an optimization-based retrieval method for handling generalized cases. It is interesting to note that the use of generalized cases can provide the added benefit of reducing the size of a case-base (compacting – removing unnecessary cases) in order to ensure better retrieval performance.
- **Graph Structures** – a retrieval method for domains in which cases are represented as graph structures. For example, Bunke and Messmer [15] propose a similarity measure that is based on a sub-graph matching algorithm, which uses graph-editing operations (i.e. insertion, deletion, node substitution, etc.).

It is worth noting that structural similarity-based approaches tend to require a hierarchical case representation. Evidently, the design choice to represent a case as either “flat” or hierarchical depends on the nature of the application domain being modeled. We shall be considering this design choice in more detail in Section 4.3.

3.4.5 Reuse and Revision Strategies

In order to simplify the following discussion, we shall assume that the intended application domains for CBR are sufficiently complex to warrant the need for an explicit revision phase. In other words, pending the retrieval phase, the most similar solution is not blindly applied, but an analysis or evaluation (i.e. possibly using a reasoning

component) is carried out to examine if case adaptation or retrofitting is required. As per Watson [113], under such circumstances the reuse and revision phases are often interpreted as a single phase – the case adaptation phase. Adaptation strategies can be classified into two main categories:

- **Substitution Adaptation** – where the adaptation activities modify some features of the retrieved solution.
- **Transformation Adaptation** – where the adaptation process alters the structure of the solution (i.e. add or remove some features).

Over the past decade, many CBR systems have demonstrated the effectiveness of performing substitution and transformational adaptation⁷. For example, a menu planning system named CHEF [39] has used substitution adaptation to replace recipe ingredients, while transformational adaptation was applied to amend associated recipe instructions to adapt meal recipes as per menu requirements. In addition, Garza and Maher [34] have proposed the use of evolutionary methods to perform both substitution (i.e. mutation operations) and transformational adaptation (i.e. crossover operations), in the context of a CBR system for proposing architectural designs.

Although CBR systems avoid reasoning from first principles (by remembering and reusing past solutions), since the task of adapting a case may be too complex for a user to undertake (i.e. a novice data miner adapting a DM case) knowledge-intensive CBR applications often carry out the case adaptation phase using supplementary “knowledge” in the form of a reasoning mechanism (i.e. rule-based inference, evolutionary computing, neural networks, etc). Hence, it is worth noting that the acquisition of this adaptation knowledge may require a substantial knowledge engineering effort. See Section 3.6 and 3.7 respectively for more details on knowledge engineering and KI-CBR.

⁷ Unfortunately, an exhaustive treatment of case adaptation strategies is beyond the scope of this document.

3.4.6 Retention and Maintenance Strategies

In recent years, the widespread application of CBR systems (i.e. the existence of large sized case-based containing thousands of cases) has encouraged research into effective case retention and maintenance strategies. The following presents some of the key ideas that have arisen from such activities.

As mentioned earlier, the *retain* phase represents the final step in the CBR life cycle where the problem-solving episode is incorporated into the system's knowledge base or case library. The traditional view for retention in CBR systems has been to simply record the target problem specification and final solution with the assumption that the outcome was successful. For simple application domains this approach may be adequate, nevertheless for domains where the outcome is less reliable or when the criteria for success is more complex, cases must contain additional information on the outcome of a particular solution (i.e. solution quality assessment, derivational traces, etc.) [114]. Hence, the modern view of retention encompasses a much broader perspective of the overall CBR learning process. For example, McSherry *et al.* [64] have proposed techniques for detecting inconsistencies prior to case retention. In addition, they have also presented a system known as *CaseMaker* that performs background reasoning to assist a user in deciding which cases are best to add during case authoring activities.

Although an exhaustive treatment of case maintenance strategies is beyond the scope of this report, in brief, case maintenance is primarily concerned with addressing CBR performance issues relating to the following :

- **Harmful cases** – These range from the simple presence of duplicate cases to the occurrence of outlier cases (i.e. spurious cases which affect overall similarity assessment) within a case library. Amongst others, Wilson and Leake [56] have proposed the periodic use of maintenance policies (i.e. case editing, case deletion, case coverage analysis, etc.) to manage harmful cases.

- **The CBR utility problem**⁸ – First demonstrated by Smyth and Cunningham [95], the CBR utility problem asserts that a trade-off exists between retrieval efficiency and adaptation efficiency as the size of a case base grows and eventually reaches a critical size.
- **Inadequate Competence or Coverage** – Is defined as the range of target problems that a CBR system is capable solving. Poor coverage occurs when cases are not sufficiently distributed (or suitably representative for the intended application domain) across the case-base problem-solution space.

To complicate matters further, it eventually became clear that the deletion of cases (to address harmful cases and the utility problem) irrevocably reduced the competence of the case base. Hence, Smith & McKenna [96] proposed the use of a competence model to evaluate the contributions of individual cases in order to guide the selection of candidate 'pivotal cases' and/or cases for deletion. On a similar note, based on their competence model, they also contributed research towards the definition of overall CBR performance metrics such as efficiency (average problem resolution time), competence and solution quality.

3.4.7 An Evaluation of Available CBR Frameworks

As indicated by the *STATUS* column of Table 1, it is clear that the vast majority of the CBR frameworks were unavailable (i.e. either the product owners did not reply to our e-mail queries for our acquiring an evaluation copy – indicated by *No Reply* - or the CBR frameworks were simply not accessible to the general public – indicated by *Unavailable*).

⁸ It is important to note that utility here applies to the notion of an efficiency trade-off and should not be confused with the traditional AI notion of utility theory (i.e. a relative measure of happiness or satisfaction).

Table 1 Summary of Evaluated CBR Frameworks

#	NAME	OWNER	STATUS
1	IUCBRF	Indiana University	Evaluated
2	Jcolibri	GAIA Group	Evaluated
3	CBR-Works	TeccInno, Germany	Unavailable
4	CBR*Tools	INRIA Group, France	No Reply
5	CREEK	Aamodt Group	No Reply
6	Oreng	TeccInno, Germany	Unavailable
7	LPA	LPA	Evaluated
8	Esteem	Esteem Software	Unavailable
9	CasePoint	Inference Corp.	No Reply
10	ART*Enterprise	BrightWare, USA	Unavailable
11	CBR-Express	Inference Corp.	No Reply
12	Remind	Cognitive Systems Inc	Unavailable
13	ReCall	ISoft	No Reply
14	KATE-CBR	AcknoSoft, France	No Reply
15	CASUEL	INRECA Project	Unavailable
16	Eclipse	Haley Enterprise	Retired
17	Caspian	University of Wales	Evaluated
18	Fionn	Trinity College, Dublin	Evaluated

Hence, our hands-on evaluation was restricted to five CBR frameworks: *IUCBRF* [44], *Jcolibri* [48], *Caspian* [18] *Fionn* [32] and *LPA* [61]. The *Caspian* product (DOS-based) was considered inadequate for our needs: it contained a proprietary case library format and provided no useful GUI or similarity measure primitives. In addition, though both the *Jcolibri* and *Fionn* frameworks seemed promising these were still considered as lacking maturity and flexibility for us to seriously consider them for our research needs. The former contained a wizard based on the Problem Solving Methods (PSM) paradigm for setting up a CBR, while the latter contained a flexible java-based library for implementing a basic CBR application. The *LPA* CBR extension product was focused more for SQL-like retrieval and did not provide any useful primitives for dealing with

similarity-based retrieval or case adaptation. Though this product might be useful for performing database-oriented searching, it was deemed inadequate for our research needs. The most promising CBR product we evaluated was the *IUCBRF* framework. Though *IUCBRF* (a java-based library for implementing a CBR applications) was fairly easy to configure for a given application, it was finally judged that the product did not offer a substantial advantage at the time (i.e. no database back-end support, basic similarity measures framework and limited GUI primitives for building the data entry components), when compared to implementing our own basic, functional and flexible CBR framework (see Section 4.3).

3.5 Ontologies

3.5.1 Definition

Though ontology definitions are numerous and varied, an early yet succinct and meaningful definition was provided by Neches *et al.* [68] as follows:

“An ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary”.

Moreover, perhaps the following more detailed explanation by Studer *et al.* [98] will further strengthen our understanding of ontologies:

“An ontology is a formal, explicit specification of a shared conceptualization. *Conceptualization* refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. *Explicit* means that the type of concepts used, and the constraints on their use are explicitly defined. *Formal* refers to the fact that the ontology should be machine-readable. *Shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private for some individual, but accepted by a group. ”

One of the key objectives for developing formal DL ontologies was to provide syntactic and semantic primitives for the declaration of non-ambiguous knowledge that is amenable to automated reasoning.

3.5.2 Some Advantages of Using Ontologies

According to Noy and McGuinness [69], the following are some of the key benefits of developing an ontology:

- **Knowledge Sharing** – Provide a common understanding or information structure amongst people or software agents. A typical example is the use of ontologies for e-commerce activities (i.e. Semantic Web). For example, pricing and product information could be represented using an ontology and shared within a network of vendors. Subsequently, autonomous agent entities could reason and make reliable purchase decisions on behalf of the vendors.
- **Foster Reuse Between Domains** – The effort involved with the development of an ontology can be very costly and time consuming. Hence, if generic ontologies are expressed in a portable and unambiguous manner, these can be shared with other organizations in related disciplines.
- **Explicit Domain Knowledge** – Unlike software development paradigms where domain knowledge is intrinsically embedded within the application, ontologies provide a flexible and structured approach for modeling domain knowledge separately from application control and logic.
- **Analyzing Domain Knowledge** – Since ontologies provide a formal specification of a domain (both syntactic and semantic), these can be very valuable for carrying out formal analysis either manually amongst experts or using automated reasoning mechanisms (i.e. classifiers, reasoners, etc.).

3.5.3 Common Applications of Ontologies

The following provides a brief classification and overview of the main types of ontologies that have arisen out of research and industry efforts [38]:

- **Knowledge Representation** – A knowledge ontology is a sort of meta-ontology that is used to express modeling primitives (i.e. class, instances, relations, attributes, etc.) for a given representation paradigm (i.e. frames, description

logics, etc.). Most ontology languages such as OCML [71], RDF(S) [84], OIL [72] and OWL [75] have associated knowledge representation ontologies.

- **Top-Level or Upper-Level** – these ontologies describe very general concepts that are common across domains and contain abstract concepts such as time, events and space. Top-level ontologies are sometimes used to construct domain ontologies, however typically completed domain ontologies are often later linked or integrated to the former. *Cyc* [26] is an example of a top-level ontology that holds a large amount of “common sense” knowledge (i.e. facts, rules of thumb, and heuristics for reasoning about the objects and events of everyday life). *SUMO* (Suggested Standard Upper Ontology) [99] is another upper-level ontology that contains comprehensive knowledge of time, plan and process theories.
- **Linguistic Ontologies** – are used to describe semantic constructs rather than to model a specific domain. They are commonly used for natural language processing and define grammatical unit concepts (i.e. words, verbs, adjectives, etc.). *Wordnet* is an example of a linguistic ontology that contains a large lexical database of the English language [119]. *WordNet* attempts to organize lexical information in terms of meaning.
- **Domain Ontologies** – Unlike other previously mentioned ontologies, domain ontologies specify the vocabularies, concepts and relations for representing a specific domain or area. Over the past decade, such ontologies have been developed for domains ranging from e-commerce and medicine to engineering, chemistry and multi-agent systems. Examples of domain ontologies are *UMLS* (Unified Medical Language System) [108] which contains clinical terminology used by medical professionals for aiding with diagnosis and *EngMATH* [30] which holds mathematical models used by engineers to analyze physical systems.

3.5.4 Ontology Markup Languages

Unfortunately, an exhaustive treatment of every available ontology language is beyond the scope of this report. Hence, we present some of the key ontology markup languages that have arisen from research in the area of the semantic web. Specifically, we shall pay particular attention to the OWL language since it has virtually become the *de facto* ontology language for the semantic web, and from the fact that it has played a pivotal role during our research and subsequent realization of an intelligent DM assistant.

- **RDF(S)** – the combined Resource Description Framework (RDF) and associated RDF vocabulary description language (RDFS), both developed by the W3C, is a data model based on the semantic network formalism that essentially permits the creation of metadata for describing web resources. The RDF data model basically consists of three components: resources, properties and statements. Resources are described by URIs, properties (or predicates) are used to define attributes (or relations) to describe a resource, while statements are used to assign a value to a property for a given resource. Although inference support is available, RDF(S) has primarily gained popularity from its use with ontology query languages such as RDQL [85] and SPARQL [97].
- **OIL** – The Ontology Inference Layer (OIL) was developed in the context of a European project. OIL is a frame-based language that uses DL to provide clear semantics and to permit the implementation of efficient and decidable decision procedures or reasoners. OIL is a SHIQ⁹ language. OIL was the first language to combine elements from frame languages and web standards such as XML and RDF.

⁹ See Table 4 in the Appendix Section for additional details on the meaning of the acronyms used to describe the SHIQ description logic.

- **DAML+OIL** – this ontology language (a successor to OIL) was developed by a joint USA and European Union committee. This language retained the DL derived language constructors of OIL, but it discarded the notion of frames in favour of DL style axioms which were more compatible with an RDF syntax.
- **OWL** – this language is the result of work from the W3C Web Ontology Working Group. This language supersedes the DAM+OIL language. OWL is intended for publishing and sharing ontologies on the Web, though it has been applied in many other areas (i.e. software engineering, content management, bioinformatics, etc.). Like DAML+OIL, OWL is built upon RDF(S). OWL uses URI (Universal Resource Identifier) for referencing names, XML Schema for supporting data values and provides import and namespace primitives for connecting documents on the World Wide Web. In trying to satisfy a large number of requirements, OWL supports three sub-languages: OWL-Lite, OWL-DL and OWL-Full. OWL-DL and OWL-Lite versions provide certain expressivity constraints in order to ensure decidability of inferences, while the OWL-Full version provides greater expressive power (at the expense of being undecidable). OWL-DL is very close to the *SHOIN(D)*¹⁰ description logic. Essentially, OWL-DL can form descriptions of classes, data types, individuals, data values using the constructs as defined in Figure 15. The first column represents the abstract syntax (similar to that which is used in OWL), the second column represents the DL syntax, while the third column indicates the formal semantic details.

It is interesting to note that the above-mentioned ontology mark-up languages, from RDF(S) to DAML+OIL, also represent the languages that have most influenced the design and evolution of the OWL ontology language [42].

¹⁰ See Table 4 in the Appendix Section for additional details on the meaning of the acronyms used to describe the *SHOIN(D)* description logic.

Abstract Syntax	DL Syntax	Semantics
Descriptions (C)		
A (URI reference)	A	$A^I \subseteq \Delta^I$
owl:Thing	\top	owl:Thing ^I = Δ^I
owl:Nothing	\perp	owl:Nothing ^I = $\{\}$
intersectionOf($C_1 C_2 \dots$)	$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = C_1^I \cap C_2^I$
unionOf($C_1 C_2 \dots$)	$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = C_1^I \cup C_2^I$
complementOf(C)	$\neg C$	$(\neg C)^I = \Delta^I \setminus C^I$
oneOf($o_1 \dots$)	$\{o_1, \dots\}$	$\{o_1, \dots\}^I = \{o_1^I, \dots\}$
restriction(R someValuesFrom(C))	$\exists R.C$	$(\exists R.C)^I = \{x \mid \exists y. (x, y) \in R^I \text{ and } y \in C^I\}$
restriction(R allValuesFrom(C))	$\forall R.C$	$(\forall R.C)^I = \{x \mid \forall y. (x, y) \in R^I \rightarrow y \in C^I\}$
restriction(R hasValue(o))	$R : o$	$(R : o)^I = \{x \mid (x, o^I) \in R^I\}$
restriction(R minCardinality(n))	$\geq n R$	$(\geq n R)^I = \{x \mid \#\{(y, (x, y) \in R^I)\} \geq n\}$
restriction(R maxCardinality(n))	$\leq n R$	$(\leq n R)^I = \{x \mid \#\{(y, (x, y) \in R^I)\} \leq n\}$
restriction(U someValuesFrom(D))	$\exists U.D$	$(\exists U.D)^I = \{x \mid \exists y. (x, y) \in U^I \text{ and } y \in D^I\}$
restriction(U allValuesFrom(D))	$\forall U.D$	$(\forall U.D)^I = \{x \mid \forall y. (x, y) \in U^I \rightarrow y \in D^I\}$
restriction(U hasValue(v))	$U : v$	$(U : v)^I = \{x \mid (x, v^I) \in U^I\}$
restriction(U minCardinality(n))	$\geq n U$	$(\geq n U)^I = \{x \mid \#\{(y, (x, y) \in U^I)\} \geq n\}$
restriction(U maxCardinality(n))	$\leq n U$	$(\leq n U)^I = \{x \mid \#\{(y, (x, y) \in U^I)\} \leq n\}$
Data Ranges (D)		
D (URI reference)	D	$D^I \subseteq \Delta_D^I$
oneOf($v_1 \dots$)	$\{v_1, \dots\}$	$\{v_1, \dots\}^I = \{v_1^I, \dots\}$
Object Properties (R)		
R (URI reference)	R	$R^I \subseteq \Delta^I \times \Delta^I$
	R^-	$(R^-)^I = (R^I)^-$
Datatype Properties (U)		
U (URI reference)	U	$U^I \subseteq \Delta^I \times \Delta_D^I$
Individuals (o)		
o (URI reference)	o	$o^I \in \Delta^I$
Data Values (v)		
v (RDF literal)	v	$v^I = v^D$

Figure 15 OWL DL Descriptions, Data Ranges Properties and Values (source [42])

3.5.5 An Evaluation of Rule-Supported Ontology Frameworks

An important aspect of our research has focused on the integration of both declarative and procedural knowledge (and associated DL-based and rule-based reasoning paradigms) within the context of formal OWL-DL ontologies. Hence, the following discussion specifically focuses on the availability and use of rule-based inference engines (i.e. expert system shells) as they apply to a plausible integration with formal DL ontologies. With respect to the following evaluation of rule-supported ontology reasoning frameworks, it is important to distinguish between a stand-alone rule-based inference engine (i.e. JESS [50]) and a framework (i.e. SWRL-Jess-Bridge [82], ROWL [87]) which provides various interfacing and translation mechanisms for interacting with a stand-alone rule-based inference engine.

Though several other less appropriate frameworks were evaluated, Table 2 illustrates the key rule-supported ontology reasoning frameworks that were investigated. Our

investigation for the best framework was mainly motivated by the following requirements:

- a) Support the integration of rules within OWL ontologies.
- b) A rule-based inference engine that supports forward-chaining.
- c) An API for exporting/importing ontology elements (i.e. facts) and rules to/from an inference engine.
- d) An API for invoking forward-chaining inference.
- e) Support for expressing rules using the SWRL specification.
- f) Integration with the Protégé ontology development environment.

Table 2 Summary of Ontology and Rule Reasoning Frameworks

#	NAME	OWNER	STATUS
1	JESS	Sandia National Laboratories	Evaluated
2	Jena	HP Labs Open Source Project	Evaluated
3	ROWL	Semantic Web Central Open Source	Evaluated
4	KAON2	Joint European Community Effort	Unsuitable
5	HOOLET	University of Manchester	Unsuitable
6	SWEETRULES	Massachusetts Institute of Technology	Unsuitable
7	SWRL-Jess-Bridge	Protégé API extension	Best Choice

Early on during our evaluation we quickly concluded that JESS was by far the most attractive stand-alone inference engine primarily because of its high performance, flexibility and popularity. The Jena toolkit [49] provides both a proprietary rule-based inference engine and a flat-file rule syntax format, however the rules cannot be explicitly integrated within an ontology (i.e. using SWRL rule specification or other method). ROWL [87] provides a simple XSLT-based framework for translating OWL ontology files for use with the JESS engine, however it does not support the SWRL rule specification. The KAON2 [54] ontology development framework provides a rule-based reasoner and some support for the SWRL rule specification. However, at the time of our evaluation we decided not to use the KOAN2 approach due to some existing shortcomings with rule specification. HOOLET [41] is an implementation of an OWL-DL reasoner that uses

the Vampire theorem prover [110]. The base implementation has been extended to support the translation of SWRL rules into FOL which can then be applied to the theorem prover. SweetRules [101] is a large open-source suite that integrates various semantic web technologies such as Jess, Jena and SWRL. Our investigation concluded that this framework was overly complex and not adequately integrated (i.e. easy to use) for our research needs. The *SWRL-Jess-Bridge* API [102] is an extension to the *Protégé development API* [82] and provides a simple programming interface for exporting/importing ontology elements (i.e. facts) and rules to/from the JESS engine. In addition, the *SWRL Rule editor* plug-in, available with the *Protégé-OWL editor* [81], was also used for eliciting rules and performing basic validation tests with the JESS reasoner. Finally, as shall be elaborated upon in the next section, the combination of the JESS [50] engine and the *SWRL-Jess-Bridge* API were chosen for our implementation since these adequately met the previously mentioned requirements (see requirements a to f).

3.5.6 Ontologies and Rule-Based Reasoning

Before engaging into the specific details about the integration of rules with ontologies, it is only fair that we briefly introduce the Semantic Web Rule Language specification (SWRL) [103], since it has virtually become the *de facto* rule language for the Semantic Web. SWRL is based on the combination of OWL-DL and OWL-lite sub-languages. SWRL allows users to write Horn-like¹¹ rules expressed in terms of OWL concepts to reason about OWL individuals. In addition, the SWRL specification does not impose restrictions on how reasoning should be performed with the rules. SWRL rules are written as antecedent-consequent pairs. SWRL provides support for variables, referring to individuals, string literals and various operators for testing relations and equality. For example, the following rule expresses that if a person (?x1) has a sibling (?x2), and that sibling is a man, the implication is that they are brothers (?x1, ?x2):

```
hasSibling(?x1, ?x2) ^ Man(?x2) -> hasBrother(?x1, ?x2)
```

¹¹ A Horn clause is a disjunction of literals with at most one positive literal or atom. Equivalently, it can be expressed as an implication statement containing a conjunction of literals on the antecedent side and a single literal on the consequent side.

In the above rule, *hasSibling* and *hasBrother* are predicates and *Man* is a concept. Evidently, if such a rule were executed by a reasoner, the consequent of the rule could affect all individuals (?x1) that satisfy the rule antecedent. Within the context of OWL-DL ontologies, Golbreich [37] first proposed a prototypical implementation of the *SWRL-Jess-tab* plugin for the Protégé OWL ontology editor [81]. This prototype provided a GUI interface for reasoning with SWRL rules combined with OWL ontologies, using the RACER [83] and JESS [50] reasoners. Shortly after, O'Connor *et al.* [70] provided a configurable and programmable interoperation environment for reasoning with SWRL rules and OWL ontologies using third party rule engines. Specifically, they provide the *Protégé SWRL Rule editor* (available within the *Protégé ontology editor*) that allows for eliciting rules which are represented as individuals within an OWL ontology. Their solution also provides a flexible java-based framework for the dynamic management of SWRL rules (i.e. creation, deletion, etc.) and the *SWRL-Jess-Bridge* API for interfacing with the JESS engine. With this API, one can export/import rules and ontology “facts”, reason and affect (update or create) new ontology individuals.

3.6 The Knowledge Engineering Process

Knowledge engineering has evolved since the late 1970s from the art of building expert systems and knowledge-based systems (a.k.a. knowledge systems). Briefly stated, knowledge is the body of data and information that people bring to bear on problems or tasks and hopefully enables their resolution (or the creation of new information or knowledge in the process). The knowledge possessed by human experts is often unstructured and difficult to express or formalize. Hence, a major goal of knowledge engineering (KE) has been to help articulate, capture, and formalize domain knowledge in a reusable form. Though many software development methodologies have been proposed over the past several decades, the following presents the key characteristics of the *CommonKADS* methodology for building knowledge systems [93].

3.6.1 Knowledge Engineering Model Types

The *CommonKADS* knowledge engineering methodology can be represented as taxonomy of models which are organized into three distinct layers. First, the *context* layer defines the following three high-level model types:

- **Organizational Model** – Used to analyze the major features of an organization such as problems, objectives, opportunities and risks.
- **Task¹² Model** – Is used to analyze the relevant tasks or sub-parts of a business process and associated inputs, outputs, resources, competence, etc.
- **Agent¹³ Model** – This model describes agent characteristics such as competence, authority and communication links.

Subsequently, the *concept* layer defines the following intermediate model types:

- **Knowledge Model** – Is used to explain the types and structures of knowledge used in performing the defined tasks. It provides an implementation independent description of the knowledge sources that will eventually be implemented and exploited within the organization (i.e. CBR, rule-based system, ontology, etc.).
- **Communication Model** – Used to describe the communication methods and transactions that are carried out by agents in order to execute tasks.

Finally, the *artefact* layer is defined by the following model:

- **Design Model** – Unlike the previously stated model types (which can be viewed as high level requirements for the realization of a knowledge system), these models provide technical system specifications such as hardware and software architectural details (i.e. procedural, relational or object-oriented models, etc.).

It is worth noting that not all model types may be required in order to successfully implement a knowledge system and that such decisions are to be determined at the discretion of the knowledge system implementers.

¹² A task is defined as a "piece of work" that is to be carried out by an agent.

¹³ An agent is defined as any human or software system that executes one or more tasks.

3.6.2 A Knowledge Engineering Process Model

As illustrated in Figure 16, the *CommonKADS* knowledge engineering process typically consists of six principle roles or actors:

- **Knowledge Specialist** – Is the human owner of knowledge or domain expertise from which the system is to be built.
- **Knowledge Analyst** – Involved with the elicitation of both domain knowledge and the formal specification of the knowledge system.
- **Knowledge-System Developer** – Is responsible for the actual design and implementation of the system.
- **Knowledge User** – Is the direct (or indirect) user of the knowledge system. The users play a vital role in ensuring that the final product meets the intended usability and operational requirements.
- **Project Manager** – Is responsible for ensuring the delivery of the knowledge system in a timely fashion, within allotted budget constraints and with the intended scope and quality requirements.
- **Knowledge Manager** – Is responsible for formulating the initial knowledge strategy at the business requirements level and subsequent management and integration of the eventually deployed knowledge system within an organizational context.

In practice, the knowledge analyst elicits knowledge from the knowledge specialists and requirements from the *knowledge user*. For large projects, one or more knowledge-system developers may be involved for the construction of the knowledge system. An individual may assume multiple roles within the process. With more substantial projects, a project manager may be involved to oversee the timely delivery and quality of the

overall project, while a knowledge manager may also be involved with the definition of the overall knowledge strategy and management of the system.

It is interesting to note that the CommonKADS methodology is generic enough to be applied independently of the domain (i.e. business, research, medical, engineering, etc.) and chosen knowledge elicitation paradigm (i.e. CBR, ontologies, rule-based systems, etc.) application domains. In Section 4, we elaborate in more details on the knowledge engineering approach that was used for the realization of our DM assistant.

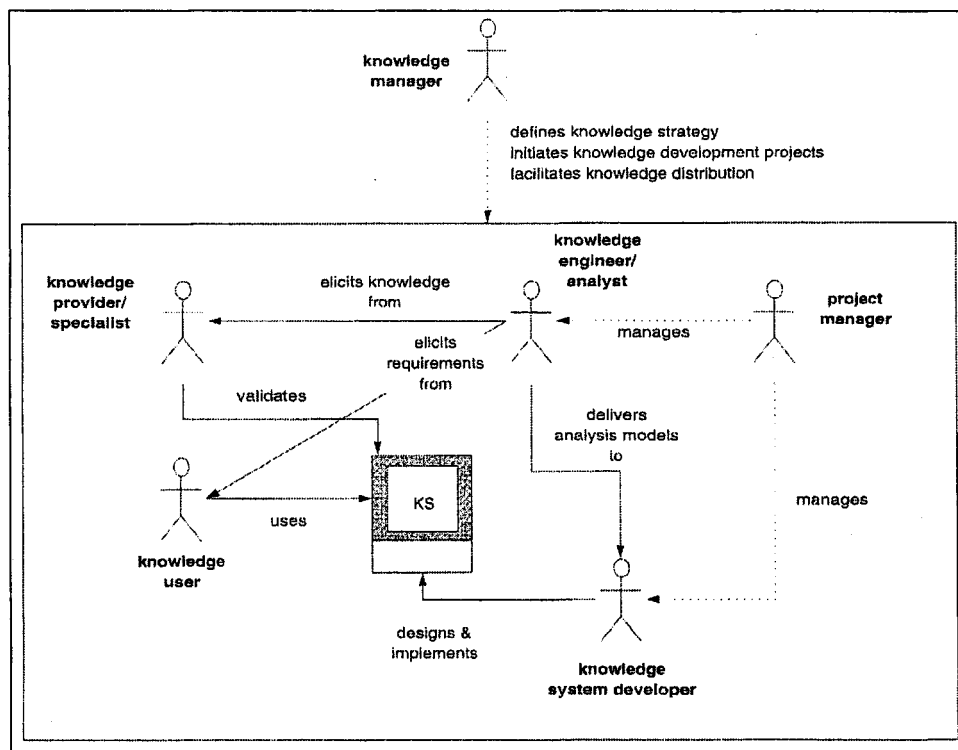


Figure 16 The Common KADS Knowledge Engineering Process (source [93])

3.7 Knowledge Intensive CBR Systems Revisited

Since the realization of our hybrid DM assistant has involved the combined use of CBR, ontology and rules knowledge representation formalisms, it seems appropriate for us close this section with a brief survey of some research activities that have combined the

CBR paradigm with a supplementary knowledge representation paradigm (i.e. ontology, rule-based expert system). Such systems are known as KI-CBR systems.

3.7.1 CBR and Rule-Based Reasoning

Several previous research efforts have demonstrated the effectiveness of combining the rule-based reasoning and case-based reasoning paradigms. For instance, An *et al.* [4] have proposed a customer relationship management application that both induces rules via case mining and makes use of a CBR to supplement the overall reasoning process when rules alone are inadequate. Marling *et al.* [63] have proposed a hybrid case-based and rule-based nutritional meal planner that is based on the CBR paradigm, yet uses a rule-base to support case adaptation by means of “what if” scenarios. From a machine learning perspective, Cercone *et al.* [20] have also demonstrated basic case and rule-based synergy by means of both a rule induction procedure (for feature weighting) to improve case retrieval and the synergistic use of induced rules to perform classification and numerical prediction. Prentzas and Hatzilygeroudis [80] have proposed a hybrid rule-based and case-based reasoning system for the medical domain that uses cases and *neurules* to perform hybrid reasoning. In addition, Montani and Bellazzi [66] have demonstrated the use of a hybrid case-based and rule-based system for diabetes prescription management. Their approach integrates cases into a rule-based reasoning framework by means of a rule refinement process.

3.7.2 CBR and Ontologies

A number of previous research efforts have demonstrated the effectiveness of combining ontologies with the CBR paradigm. Aamodt *et al.* [2] have developed a KI-CBR system called *CREEK* that is based on the use of ontologies. The *CREEK* framework strives to “enrich” cases using additional general domain knowledge represented in an ontology. Moreover, Bello-Thomas *et al.* [7] have developed a framework named *Jcolibri* for building CBR systems that use a task/method ontology (named *CBROnto*) for promoting problem solving methods re-use. Last, Bichindaritz [12] has demonstrated the use of ontologies for facilitating case structuring and acquisition.

Having surveyed the state-of-the-art of the key research domains that have most influenced our DM assistance research (i.e. DSS, CRISP-DM, meta-learning, CBR, ontologies and rules and KI-CBR), the following chapter elaborates on the specific details and design choices that were made in order to realize our novel, hybrid intelligent DM assistant.

CHAPTER 4

The Proposed Intelligent Data Mining Assistant

This chapter presents the key design considerations and implementation issues for the realization of a hybrid intelligent data mining assistant. We begin by discussing the meta-learning and CRISP-DM methodology ideas that have influenced the design of our DM assistant. Subsequently, we address important design details that were considered for implementing the core of the system - the CBR and DM ontology subsystems. Last, we present how the synergistic combination of both knowledge representation formalisms can effectively support novice data miners for carrying out DM tasks.

The early stages of our research were driven by an abstract system view or layer-cake architecture as illustrated in Figure 17:

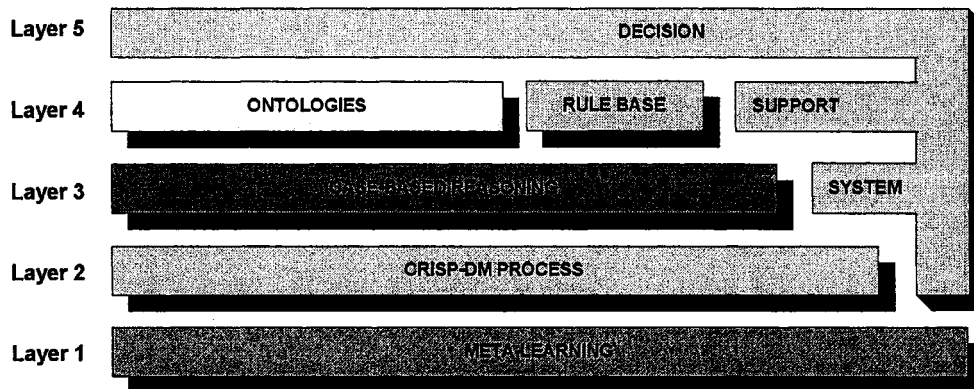


Figure 17 Architectural Overview of Intelligent Data Mining Assistant (source [23])

This abstract model is useful for highlighting the progressive, layered evolution that guided the realization of our intelligent DM assistant. First, the basis of our design was motivated by several of the aforementioned meta-learning objectives such as model selection assistance and the learning-to-learn paradigm (see Layer 1). Second, by offering a preliminary DM "knowledge vocabulary" for the realization of the CBR component, the CRISP-DM methodology has played an instrumental role in our research (see Layer 2). Armed with this solid foundation, the use of a CBR (Layer 3), a

formal ontology (integrating a rule-base) and associated reasoning mechanisms (Layer 4) provide the core knowledge components that were essential for achieving some of the DM problems and challenges stated in Section 2.1. Of utmost importance from a DM user's perspective (see Layer 5) is the fact that all the lower foundational layers should produce a useful and effective DM assistant capable of empowering a novice data miner throughout the key phases of a DM task or project.

4.1 Grounded in Meta-Learning

As previously discussed in Section 3.3.2.1 (Learning at the Meta-Level), the process of providing DM assistance can be viewed as a meta-learning problem. One of the key objectives of meta-learning has been to build meta-classifiers that are capable of effectively mapping DM datasets to models. However, we believe that DM assistance should support beyond model selection. Hence, our focus has been to extend the meta-learning problem so as to encompass both larger meta-problem and meta-solution spaces. For instance, we have added several additional problem description attributes in addition to data characteristic measures (to be discussed shortly) such as *Business Area*, *Missing Values Ratio*, and the presence of *Outlier Values*. Essentially, the decision to introduce these attributes was motivated by an interest to provide additional discriminating meta-data for "learning" a more elaborate solution space to extend the range of support when solving DM problems (i.e. an ensemble of DM solution attributes instead of single model selection attribute). For instance, the solution part can provide additional assistance for managing data preparation activities (i.e. how to handle missing, outlier or inconsistent values). Hence, as illustrated in Figure 18, we are interested in mapping DM problems to entire DM cases.

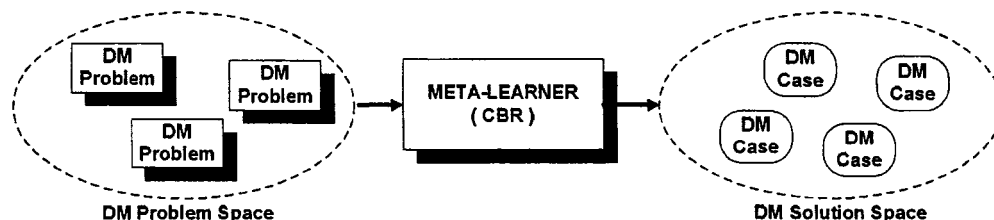


Figure 18 A Data Mining Meta-Learning Problem (source [22])

Extending the meta-learning problem beyond model selection support is not without its share of complications. For example, when compared to a simple classification problem such as a spam e-mail classifier where the solution space only contains a boolean value (true/false), our meta-learning system is viewed as having a more complex, multi-dimensional solution space. In actual fact, our DM case contains a total of 31 solution attributes (see 4.3.2 for more details). Invariably such a decision to extend the solution space favors the chances that a retrieved DM basis case using the CBR may be far from a "perfect" match to the DM problem at hand. Hence, a trade-off exists between the level of solution complexity afforded by a DM case representation and the potential case adaptation effort required to resolve a given DM task. In order to counter this effect, we have proposed the use of a supplementary knowledge source, a DM ontology containing both declarative and procedural knowledge, for supporting the subsequent case adaptation process.

As previously mentioned, the field of meta-learning has also contributed significantly in the area and use of data characterization techniques such as general, statistical and information-theoretic measures. Though we shall be elaborating in more detail on the specific problem characteristics used for our implementation (see Section 4.3.3), for the moment it suffices to emphasize that, in the context of meta-learning, the use of data characteristics have played a key role in the implementation of a suitable similarity measure for the retrieval component of our CBR system (see Section 4.3.4).

4.2 A CRISP-DM Driven Process

The CRISP-DM methodology proved indispensable in eliciting a case vocabulary for the implementation of both the CBR component. Moreover, CRISP-DM also proved equally useful for eliciting the preliminary knowledge for the DM ontology subsystem of our DM assistant.

Considering that our objective is to support the user beyond model selection, it seemed quite natural for us to integrate a DM methodology as a basis for the case structure and vocabulary. Hence, we chose to use the CRISP-DM data mining process as a basis for eliciting a set of representative attributes for our DM case representation. CRISP-DM

efficiently captures “knowledge” (in the form of a series of well defined and generalized phases, tasks and activities) of the entire data mining effort. From this, we were able to define a case representation consisting of 66 features. The problem portion of our case consists of 30 attributes, the solution portion defines 31 attributes, and the case outcome section holds 5 attributes.

Although the entire CRISP-DM methodology consists of 6 phases, the implementation of our DM assistant was restricted to the first 5 phases, namely the *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling* and *Process Evaluation* phases. Our decision for omitting the last phase (*Deployment*) was mainly motivated by the fact that most of the useful DM “knowledge” (from a user’s perspective) is concentrated within the previously mentioned 5 phases. In fact, as shall be elaborated upon later in Section 4.4 (The Ontology Component), the bulk of the “intelligent” assistance provided by our system is concentrated within the *Data Understanding*, *Data Preparation* and *Modeling* phases of the CRISP-DM methodology.

Our use of the CRISP-DM methodology was motivated by the following requirements:

- **DM Case Representation** - Acquire a knowledge source for eliciting a DM case representation (see Section 4.3.2).
- **Promote a Structured Approach** - Integrate a high-level methodology in order to foster a structured and systematic approach to the overall DM process from the user’s view point.
- **Foster Detailed DM Knowledge** - Acquire a case vocabulary that is conducive to parameterization¹⁴ in order to facilitate the elicitation of detailed DM knowledge in the form of rules (see Section 4.4.1).

¹⁴ Parameterization implies the definition of attributes that can assume a set of constrained values or enumeration (i.e. red, blue, green), instead of free form text fields (which possess very little structure).

4.3 CBR at the Core

The effective realization of a CBR-based data mining assistant involves a host of important design considerations ranging from case representation, similarity measures, indexing and feature weighting techniques to revision strategies, seed case elicitation and case maintenance issues. This section attempts to provide a concise, yet accurate presentation of how we addressed these particular CBR development issues.

4.3.1 The Key Components of a CBR System

Richter [86] has proposed an interesting way of viewing the construction of a CBR system. Essentially, the “total knowledge” within a CBR system is distributed over the following 4 knowledge containers:

- **Vocabulary Knowledge** – Provides the basic elements used to represent knowledge (i.e. predicates, attribute-value pairs, operators, functions, etc.).
- **Similarity Measure Knowledge** – Defines how to assess the “closeness” or proximity between pairs of problem descriptions or cases.
- **Adaptation Knowledge** – Contains knowledge on how a retrieved similar solution can be adequately transformed to address the given problem.
- **Case Base Knowledge** – Contains the accumulation of “experiences” or cases for a given application domain.

Figure 19 illustrates these 4 containers and the fact that the *Vocabulary* container is a fundamental or “basic” container that is solicited by the remaining three containers (i.e. *Similarity Measure*, *Case Adaptation* and *Case Base*). Over time as the CBR system accumulates experiences, these are stored in the *Case Base* container. This container is typically initialized with a set of representative “seed cases” prior to deploying the CBR application (see Section 4.3.7). The *Similarity Measure* container holds knowledge typically in the form of a similarity measure that can be used for evaluating the

“closeness” between a pair of cases and subsequent retrieval of useful cases. This knowledge can be defined using a host of previously discussed surface or structure similarity techniques. Last, the *Adaptation Knowledge* container can be represented using numerous substitution or transformational adaptation techniques ranging from the use of rule-based, ontology-based to the application of fuzzy, neural network and evolutionary computing techniques.

The construction of a CBR system involves the careful selection and distribution of domain knowledge across all 4 knowledge containers. Depending on the chosen application domain, differing knowledge engineering efforts are required in order to adequately specify the knowledge containers. Typically, the *Similarity measures* and *Adaptation knowledge* containers are the most challenging to implement from a technical perspective. Interestingly, according to Richter, knowledge can be relocated within the various containers by means of a compilation process. Moreover, it is theoretically possible (though not necessarily desired) to place all the domain knowledge within a single container. For example, if a case base of infinite size were available, it could resolve all possible domain problems. Hence, under such conditions, no similarity-based retrieval or adaptation knowledge would be required. However, the *Vocabulary* container must implicitly or explicitly exist for any given CBR application.

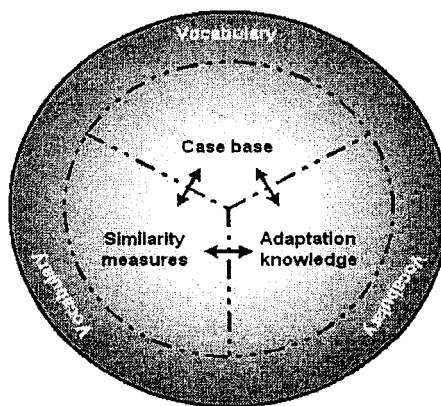


Figure 19 The View of CBR via Knowledge Containers (source [86])

4.3.2 Case Vocabulary Elicitation

As previously mentioned, the CRISP-DM methodology provided a generic knowledge model from which we elicited a preliminary set of DM case attributes or features. Subsequently, as actual DM trials were carried out with the CBR subsystem, our case representation was refined by adding and/or removing case attributes. For example, specific data characteristics (also DM case attributes) were defined based on knowledge from the area of meta-learning in general and by specific design choices made during the implementation of our similarity-based retrieval mechanism (see Sections 4.3.3 and 4.3.4). Furthermore, though we have tried to elicit a set of DM case features that are as generic as possible, our dominant use of the Weka Data Mining toolkit [115] during our DM trials has had a direct impact on the final DM case structure. For example, in the *Data Preparation* phase, the *Feature Reduction Method* attribute can only assume the following values: *Subset Selection*, *Feature Ranking*, *Principal Component Analysis* or *Singular Value Decomposition*¹⁵. The following highlights some of the key DM case attributes that were defined. For an unabridged list of all 66 case attributes, see Appendix C.

Business Understanding Phase:

- **Business Problem** – Describes the customer's primary objective from a business perspective using an informal textual description.
- **Data Mining Problem** – Used to define the criteria for the successful outcome of a DM project in technical terms (i.e. expected error rate).
- **Data Mining Activity Type** – Represents the particular DM task type such as classification, regression, clustering or association mining. For the moment, our DM assistant only supports the classification¹⁶ task.

¹⁵ The actual purpose and meaning of these possible values is beyond the scope of this report, however we shall be addressing some of these in the context of our system evaluation (see Section 5).

¹⁶ We have chosen to designate 'classification' in a general sense where it equally implies regression tasks.

Data Understanding Phase:¹⁷

- **Number of Attributes** – The quantity of attributes (or columns) for the chosen problem data set.
- **Ratio of Duplicates** – The quantity of times a given record (or example) within the problem dataset is an exact duplicate.
- **Has Outlier Values** – A boolean value to assess if the problem dataset has outlier values.

Data Preparation Phase:

- **Examples Reduction Method** – under certain situations, the sheer quantity of examples within the problem dataset may be too large for a DM algorithm. Hence, this feature is used to specify an example reduction method such as *random* or *stratified* sampling.
- **Outliers Handling** – Used to identify a method for handling outlier values within the problem dataset. Possible values are *correct*, *eliminate*, *ignore*, *estimateUsingMean*, etc.
- **Data Transformation Method** – Certain DM algorithms may require that the attributes or target concept be of a certain type (i.e. nominal or numerical). This attribute is used to specify a chosen transformation such as *binarization*, *discretization* or *normalization* for specific attributes within the problem dataset.

Modeling Phase:

- **Selected Model** – Used to define the chosen DM algorithm for producing a model. Some of the supported model types are *ID3*, *J48*, *NaiveBayes*, *LinearRegression*, and *SMO*.

¹⁷ Particularly for this phase, the features presented here are also re-introduced in the next Section (4.3.4) as feature indexes.

- **Training and Test Options** – Used to specify the chosen training and test method. Some of the permissible values are the *Bootstrap*, *10-fold Cross Validation* and the *LeaveOneOut* method.
- **Estimated Model Accuracy** – Represents the obtained model accuracy or error rate. This value can assume a value between 0 and 100 %, though our DM assistant will recommend the use of an acceptable range between 1 and 30 %.

Process Evaluation Phase:

- **Business Success Criteria Achieved** – Used for rating the level at which the business success criteria were achieved. Permissible values are *Poor*, *Fair*, *Undecided* and *Strong*.
- **DM Local Utility Score** – Used by the user for rating the level of satisfaction with respect to the overall assistance obtained by the system for the Modeling phase.

4.3.3 Problem Characterization and Feature Indexes

In order to successfully retrieve similar cases from a case base, key discriminating case attributes must be defined (a subset from the attributes elicited in the previous section) and an associated retrieval mechanism must be implemented (i.e. k-nearest neighbor). The following presents the key representative features (indexes) and the next section shall discuss how these were ultimately used to craft a global similarity measure in order to compare a problem DM case with those in the case base.

Essentially, the first step in the case retrieval process involves a DM problem characterization step. This step consists of computing the problem characteristics (a.k.a. feature indexes) for the current DM problem and subsequently comparing this “meta-data” with previously resolved DM cases in the case base, in order to find a suitable or similar matching DM case. The following describes the feature indexes that were used for characterizing a DM problem:

- **Business Area** – The business area attribute is used to discriminate amongst a finite possible set of data mining application areas within a given domain (i.e. engineering, finance, marketing, etc.). In the current context of applying data mining within a university setting, the following three typical business areas are available: (1) admission - this sub-area should be selected for analyzing data that is gathered from the admission process; (2) retention - this sub-area should be selected for carrying out data mining activities on student data corpus that is collected during regular operations; (3) follow-up - this sub-area should be selected for analyzing data that is collected from various surveys (i.e. ICOPE).
- **Number of Examples** – the quantity of observations or examples within the chosen problem dataset.
- **Number of Attributes** – the quantity of attributes (or columns) within the chosen problem dataset.
- **Number of Classes** – the quantity of classes or permissible values for a given target concept.
- **Ratio of Symbolic Attributes** – the ratio of attributes which are symbolic (nominal) over the entire attribute count of a given problem dataset.
- **Mean Skewness** – measures the symmetry of a distribution of values. Negative skewness implies a left shift, while positive skewness implies a right shifting of the distribution. The 'mean' qualifier implies that the kurtosis values are averaged over a set of attributes.
- **Mean Kurtosis** – measures the "peakedness" of a distribution of values. A higher kurtosis implies that more of the variance is due to infrequent extreme deviations as opposed to frequent modestly-sized deviations such as the shape of a normal distribution (in such a case, kurtosis is 0). The 'mean' qualifier implies that the kurtosis values are averaged over a set of attributes.

- **Normalized Class Entropy** – indicates how much “information” is necessary to specify one class value for a given target concept. The ‘normalized’ qualifier establishes a limiting range between 0 and 1.
- **Maximum Mutual Information** – is a measure of the common information shared between an attribute and the target class. The ‘maximum’ qualifier establishes that the maximum value is used for the given set of matched attribute and target class mutual information values have been computed.
- **Target Data Type** – the data type (nominal or numerical) for the chosen target concept in the chosen problem dataset.
- **Ratio of Duplicate Examples** – the quantity of times a given record (or example) within the problem dataset is an exact duplicate.
- **Has Outlier Values** – a boolean value to assess if the problem data set has outlier values or not.
- **Ratio of Missing Values** – quantity of times a given record (or example) within the problem dataset contains a missing value. A missing value qualifies if it is a NULL, SPACE or empty string (“”).
- **Has Inconsistent Values** – used to identify if the problem dataset contains inconsistent values. Data can often contain inconsistent values resulting from various data collection errors (i.e. equipment failure, data entry error, etc.). For example, an inconsistent value can result from an incomplete telephone number, ZIP code or a non-permissible value (i.e. entering a negative value for an age or height attribute).

See Appendix B for permitted value ranges and mathematical formulas (where applicable) for computing the above mentioned DM problem characteristics.

We elicited a total of 14 indexes from 5 different “discriminating” areas. The discriminating areas are illustrated in Figure 20 (along with their respective counts). For instance, the *Business area* index is used to discriminate amongst a possible finite set of application areas within a given organization (i.e. engineering, finance, marketing, etc.). In addition, we implemented a “core” subset of the feature indexes as data characteristics which are commonly available from the area of meta-learning (indicated by General, Statistical and Info-Theory in Figure 20). Moreover, we introduced four additional indexes (i.e. *Ratio of Duplicates*, *Has Outlier Values*, *Ratio of Missing Values* and *Has Inconsistent Values*) since these provide discriminating power for representing DM cases, where valuable knowledge concerning data quality and data preparation can later be solicited by the system.

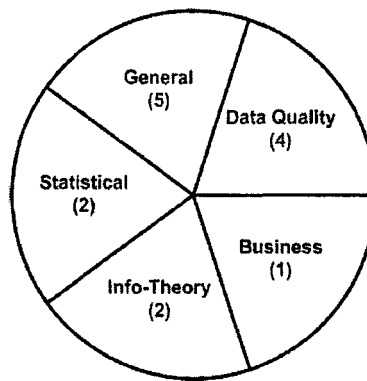


Figure 20 The Distribution of Feature Index Types

4.3.3.1 Data Characterization and Incomplete Data Sets

Prior to engaging data modeling activities, a data miner must deal with data quality issues (i.e. missing, incomplete, invalid and outlier values). Failing to address data quality issues within the initial problem set results in poor model quality (i.e. poor error rate). Similarly, during our initial trials with the CBR sub-system, we encountered difficulties during case retrieval when certain feature indexes (i.e. mean kurtosis, mean skewness, normalized class entropy and mutual information) could not adequately be computed depending on the “quality” of the problem data set. Evidently, it is unrealistic to assume that a problem dataset does not contain missing values. For example, if a particular attribute contains missing values, the skewness data characteristics cannot be computed. Simply omitting this meta-data (index feature) during the case retrieval

phase significantly affects the CBR's retrieval accuracy. Hence, in order to mitigate this problem, we implemented a simple filter that temporarily replaces the missing values with mean (numerical attribute) and mode (nominal attribute) values and computes the problem dataset's data characteristics. Our approach was based on the "ignorability" or "Missing at Random" (MAR) hypothesis as described by Schafer [91]. It is worth noting that within our context a "missing value" can take the form of a database NULL, SPACE, or empty string ("").

4.3.4 Similarity and Utility-Oriented Retrieval

The following elaborates on specific design issues that were considered during the implementation of the case retrieval mechanism for our DM assistant, and particularly our proposal for a new approach for improving retrieval accuracy based on utility theory.

4.3.4.1 The Similarity Measures

Global Similarity Measure (GSM):

The case base reasoning component of our DM assistant was initially implemented using a K-nearest neighbour classifier and the following feature-weighted, global similarity measure:

$$GSM(q, c) = \frac{\sum_{i=1}^N \delta_i w_i sim_i(q, c)}{\sum_{i=1}^N \delta_i} \quad (1)$$

A GSM consists of a sum of local similarity measures (to be discussed shortly) and assumes a bounded value between 0 and 1. The discriminating parameter, feature weights and local similarity measures are respectively indicated by the δ , w_i and $sim_i(q, c)$ symbols in Equation (1). Our choice for using the nearest neighbour was mainly motivated by its simplicity, flexibility (low maintenance for adding new cases), reasonable performance, and by the fact that we initially had few recorded DM cases available.

Depending on whether a given DM problem dataset contains only numerical or nominal values, the statistical or information-theoretic index values may or may not contribute to the problem characterization. Hence, the δ parameter is used (by providing a “non-applicability” condition) to handle potentially missing index values and prevents the global similarity measure from biasing. In addition, the δ parameter was also used to maintain a good proximity measure for asymmetric boolean indexing features (such as *Has outliers* and *Has Inconsistents* features) [104].

Local Similarity Measures and Weights:

As indicated in Table 3, the index data types are categorized either as *Nominal*, *Bounded Numerical* or *Numerical*. The *Nominal* data type can assume a finite set of elements (i.e. true, false). The *Bounded Numerical* types can assume an infinite set of values within the range of real numbers; however these are constrained within a certain range (i.e. between 0 and 1). Last, the *Numerical* data types are those which can theoretically span the entire set of real numbers (unbounded). For our implementation, we chose to use the following “exact match” equation for the *Nominal* type indexes:

$$s = \begin{cases} 1 \rightarrow x = y \\ 0 \rightarrow x \neq y \end{cases} \quad (2)$$

Essentially, this equation states that if index values are an exact nominal match, the LSM is 1, otherwise LSM equals 0. In addition, we chose to use the following simple similarity measure for both *Bounded Numerical* and *Numerical* index types:

$$s = 1 - \frac{|x - y|}{\max - \min} \quad (3)$$

This equation calculates the absolute numerical difference between two attribute values and divides this “dissimilarity” result by a maximum operating range (essentially, normalizing the result between 0 and 1). The dissimilarity value is then complemented to obtain a final local similarity measure. The key to using Equation 3 was choosing a suitable range (i.e. *min* and *max* values) for the *Numerical* index types. The choice was easy for the *Bounded Numerical* types which assumed a well defined range between 0 and 1. However, for the *Numerical* types which can assume values up to infinity (i.e. *No. Examples*, *No. Attributes*, *No. Classes*, *Mean Skewness* and *Mean Kurtosis*), an

acceptable range had to be defined. Though we could have employed more sophisticated non-linear local similarity measures for dealing with such data types (i.e. using logarithmic or exponential factors), we opted for a more simple approach. The ranges (*min* and *max* values) were established following a statistical analysis for a constrained set of problem datasets as discussed in Section 4.3.7 (Eliciting Seed Cases).

Table 3 Local Similarity Measures and Weights

#	Similarity Measure	Data Type	LSM	Weight
1	Business Area	Nominal	See Equation 2	0.6
2	No. Examples	Numerical	See Equation 3	0.6
3	No. Attributes	Numerical	See Equation 3	0.6
4	No. Classes	Numerical	See Equation 3	0.6
5	Symbolic Attributes Ratio	Bounded Numerical	See Equation 3	0.6
6	Mean Skewness	Numerical	See Equation 3	2
7	Mean Kurtosis	Numerical	See Equation 3	2
8	Norm. Class Entropy	Bounded Numerical	See Equation 3	2
9	Max. Mutual Information	Bounded Numerical	See Equation 3	2
10	Target Data Type	Nominal	See Equation 2	1
11	Duplicates Ratio	Bounded Numerical	See Equation 3	0.5
12	Has Outliers	Nominal	See Equation 2	0.5
13	Missing Values Ratio	Bounded Numerical	See Equation 3	0.5
14	Has Inconsistents	Nominal	See Equation 2	0.5

In order to improve on retrieval accuracy for k-NN based approaches, it is customary for a weight to be assigned for every feature index to reduce the influence of redundant and irrelevant features. Under an ideal setting, where ample historic DM cases were available, an effective approach for establishing weights is via the use of machine learning techniques [117]. Unfortunately, due to limited historical DM case information, an expert committee (to be addressed shortly) was consulted in order to provide a preliminary set of index weights. During DM trials, these weights were subsequently adjusted and the final values can be viewed in Table 3. In brief, the strongest weights

were assigned for the data characteristic measures, *Business Area* and *Target Data Type* indexes.

4.3.4.2 The Utility Measures

From a theoretical perspective, Bergman *et al.* [9] were the first to propose the possibility of extending the traditional CBR with a utility-oriented approach. Inspired by this idea, during early experimentation we quickly ascertained that the final decision to select the most “appropriate” case was not obvious (strictly based on a GSM and the k-NN classifier). For example, since the solution part of a DM case is multi-dimensional (i.e. advice for data quality, data preparation, modeling parameters, model evaluation, etc.), a user may have a preference for using a previous DM case that offers more data preparation support (over data modeling information). In addition, different DM cases hold various levels of “solution quality” depending on how each user has approached the problem. This kind of tacit knowledge embedded within the case is hard to assess merely by the use of a global similarity measure. Hence, in order to improve retrieval accuracy, we opted for defining a global utility measure (GUM) for each retained DM case. Even though this utility measure is somewhat subjective (it is based on the user’s level of satisfaction at each phase of the DM process for a previously resolved case), it can provide a significant refinement (for improving overall case retrieval accuracy) over a purely similarity-based retrieval approach. For instance, when presented with candidate cases, the user can evaluate the trade-off between the “similarity” of a case and the associated level of “usefulness” for a given case. Hence, a more informed decision concerning which case is the most appropriate basis case can be made (upon which adaptations can be affected to resolve a target problem).

Although utility theory has its roots in economics (where money is a common utility measure), utility measures have been effectively applied in many areas of AI for supporting decision-making [88]. A utility function maps a “state” onto a real number (i.e. typically the range of [0,1]), which describes the associated degree of “happiness” or usefulness for achieving the state. Specifically, our implementation of a GUM as indicated by Equation (4), implies the sum of three local utility measures; local utility

measures for the data understanding, data preparation, and data modeling CRISP-DM phases.

$$GUM(X) = \frac{\sum_{i=1}^3 k_i lum_i(x_i)}{\sum_{i=1}^3 \max(x_i)} \quad (4)$$

The state attributes (user satisfaction levels for a given DM phase), utility weights and local utility measures are respectively represented by the vector X , k_i and $lum_i(x_i)$ in the above equation. The denominator is used to normalize the results between 0 and 1.

Local Utility Measures and Weights:

In a similar fashion to the *ad hoc* selection process used for LSM weighting, the local utility measure weights were fine tuned during DM trials and are defined in Table 4. Table 4 illustrates that the DP and DM phases are weighted more strongly than the DU Phase. This can be explained by the fact that most of the DM detailed knowledge is concentrated within the DP and DM phases (see Section 4.4), while the DU phase is partially automated via the use of a data characterization module which automatically computes indexes values such as the Missing Values Ratio, Mean Kurtosis, etc.

Table 4 Local Utility Measures and Weights

#	Utility Measure	Weight
1	Data Understanding (DU)	0.2
2	Data Preparation (DP)	0.5
3	Data Modeling (DM)	0.3

The following example clarifies how the combined similarity and utility measures can empower a user to select the most appropriate case for carrying a DM activity forward (subsequent reuse and revision phases of the CBR process). The results illustrated in Table 5 assume that a problem description has been posed to our DM assistant and the following retrieval results are obtained. The similar cases are ranked according to similarity (GSM), however the third column provides additional utility information (GUM).

For a novice data miner, the most appropriate basis case would be to aim for a case with both a relatively strong similarity (though not necessarily the highest as in case #1), and a strong utility measure (as is shown for case #2). Though this choice is not guaranteed (on the premise that GUM is partly subjective), it provides somewhat of a measure as to the potential usefulness (i.e. adaptation knowledge) that will be available for the following case adaptation process.

Table 5 Similarity and Utility-Oriented Retrieval Example

#	Case Name	GSM	GUM
1	Student Graduation Predictor	0.921	0.727
2	<i>Student Demography Classifier</i>	0.882	0.922
3	Higher Education Classifier	0.827	0.582
4	Graduation Predictor	0.712	0.607
5	GPA vs. College Correlation	0.620	0.843

Although at the moment our GUM is deterministic (weight coefficients are used and not probabilities), it is foreseeable that future research could allow us to assign a probabilistic "confidence measure" for the specified local utilities for each case stored. In other words, a quality measure could be assigned to a utility based on a data miner's past experience for resolving "high quality" cases. Furthermore, the primary motivation for our introducing a utility measure to refine case retrieval was based on our inability to use traditional information retrieval metrics such as recall and precision for evaluating our system. For instance, recall was a moot point since we are using a fixed 5-NN retrieval method. In addition, the notion of retrieval precision was intractable since our case representation is complex and multi-classed (classifying DM case solutions is a non-trivial task), unlike a simple classification or regression problem where the class label is discernable.

Interestingly, during early research for defining the most appropriate case structure for our DM case (i.e. flat or hierarchical), we recognized that splitting our case representation into distinct sub-cases (i.e. a hierarchical case representation that uses sub-cases to represent the 5 distinct phases of the CRISP-DM process), could pose

serious problems later on for retrieving and re-constructing a coherent and similar DM case for the user. Surprisingly, due to the addition of distinct local utility measures for each DM phase, this case structure “decomposition” has resurfaced from a utility-oriented perspective.

4.3.5 Reuse and Revision Strategy

With respect to the reuse phase of our implementation, this phase implies a sort of commitment on the user’s part. For example, at this point in the CBR process, it is assumed that the user has chosen one of the proposed “similar” basis cases to work with and she will attempt to adapt this DM case (using our DM assistant and experimentation via the use of a DM toolkit) and hopefully resolve a current DM problem. Hence, once the reuse phase has been initiated by the user, the DM problem characteristics (used for DM case retrieval) are automatically transposed over the chosen basis case’s problem part. Evidently, the subsequent revision phases may imply modifications to the “solution part” of a DM case. However, this transposition ensures that the DM case correctly reflects the current problem that will eventually be resolved (using parts of a previously resolved case and the necessary adaptations to the solution part of the problem case) prior to the retain phase.

As previously noted, DM methodologies such as CRISP-DM adequately specify the phases, tasks and activities that need to be carried out during a DM project, but provide very little detailed knowledge for the novice miner on how to actually carry out a given step. Hence, for complex application domains such as DM assistance, where detailed domain knowledge may be required to decide on the appropriate choice for a given case attribute (and its potential impact on other attributes), it is fair to put forward that achieving case adaptation support invariably requires a complementary knowledge source (i.e. a DM ontology containing both declarative and procedural knowledge).

For example, the proper application of a simple linear regression model often requires that the user possess detailed knowledge for effectively carrying out the model evaluation phase for a given DM task (i.e. significance testing, residue normality and model variance verification). As such, we have proposed the use of a DL-based

ontology (combining both declarative and procedural knowledge in the form of SWRL rules) and accompanying rule-based reasoner in order to support the CBR reuse and revision phases (by providing recommendations and heuristics when possible). See Sections 4.4 and 4.5 for details on how the ontology subsystem was used for implementing case adaptation support using SWRL rules and rule-based reasoning.

4.3.6 Case Retention

Though sophisticated case retention policies have been proposed by Leake *et al.* [56], we have opted for the following simple case retention mechanism. When a case is retained within the case library it is initially defined with a 'non-certified' status or label. A *non-certified* case is one which has not been officially examined by a committee of experts in order to ensure that it does not potentially affect the overall performance of the case base (i.e. a harmful case – see Section 3.4.6). The importance of this retention mechanism shall be further addressed shortly when we cover case maintenance issues (Section 4.3.8).

4.3.7 Eliciting Seed Cases

Our primary concern during the authoring of initial “seed cases” was to ensure reasonably good case competence (i.e. coverage of the target problem space) over a constrained problem space. Although novel approaches have been proposed for authoring cases to ensure sufficient competence and performance, most of these approaches make strong simplifying assumptions (i.e. large quantities of available cases, low problem description index feature dimension, representative-ness assumption, etc.) about the application domain. Essentially, a preliminary statistical analysis (i.e. means, modes and standard deviations of indexes) was performed on real datasets provided by the UQTR decision support department. Figure 21 illustrates the estimated ranges, means or modes (indicated by the star symbol) obtained from our analysis of the indexes. From this seed DM activities were carried out in the vicinity of these established feature means/modes (i.e. a constrained area of the problem space). Afterwards, these DM cases were individually reviewed by the expert committee on simple criteria (i.e. relevance and solution quality) and officially “certified” as DM cases.

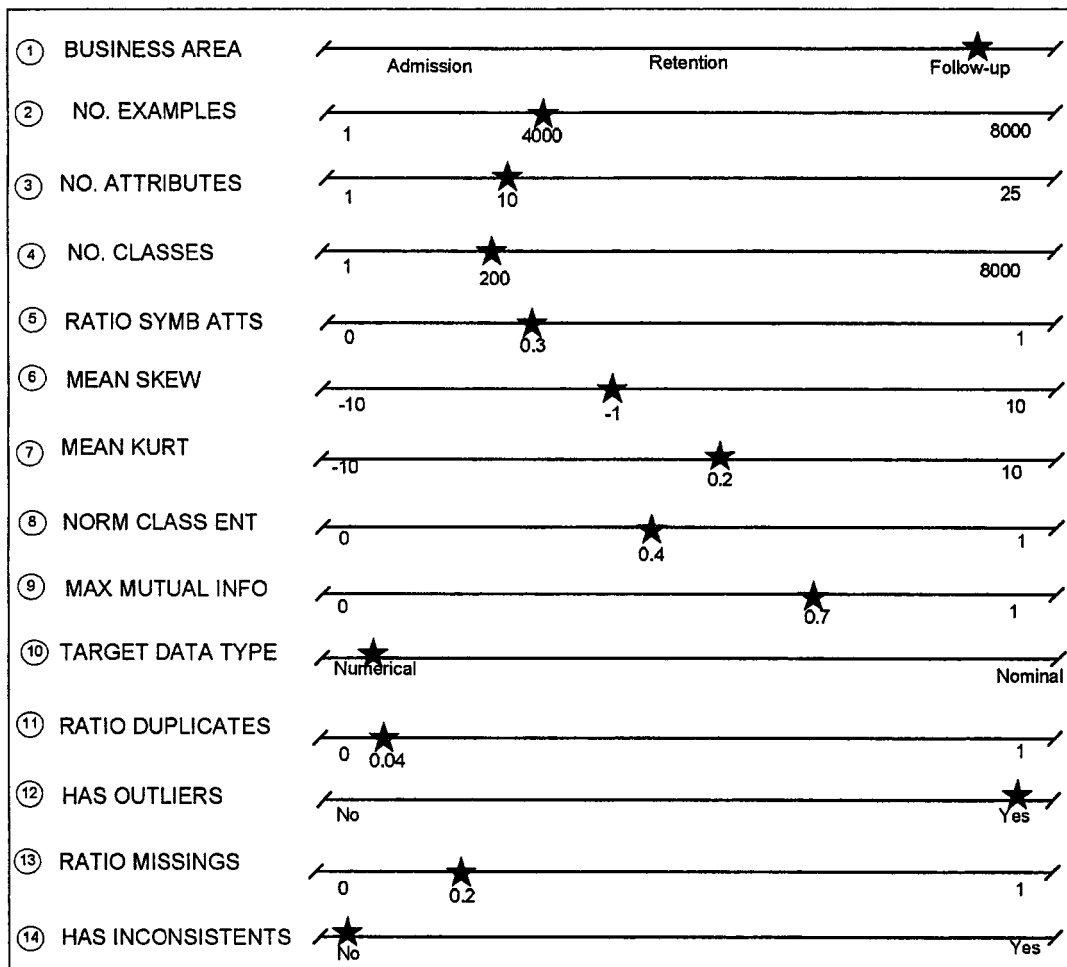


Figure 21 Index Means/Modes from Statistical Analysis of Domain Data

4.3.8 Case Maintenance Issues

Smyth and McKenna [96] have defined three types of CBR performance metrics: *efficiency*, *competence* and *solution quality*. For the moment, since our case-base is of a modest size, efficiency (problem solving time) shall not be a concern. Moreover, though Smyth and McKenna have also proposed approaches for defining simple case base competence models, due to the inherent complexities associated with DM cases (i.e. multi-dimensional solution space), we faced difficulties with formalizing the notions of *competence* and *solution quality*. Hence, our approach mainly consisted of periodically consulting a committee of experts (comprising of a statistician, domain

analyst and DM expert) to evaluate possible outlier cases, duplicate cases and take corrective actions to ensure “qualitative” case competence and solution quality.

4.4 The OWL-DL Ontology Component

In this section we present the ontology knowledge elicitation strategy used, the rule-based inference mechanism employed, as well as the resulting ontology-guided DM assistance that is much needed for supporting the case adaptation process. In brief, we have attempted to enrich our DM assistant with complementary knowledge (OWL ontology concepts, individuals and rules) in order to provide the user with adaptation or validation knowledge to complete her DM task.

During our early attempts at soliciting detailed DM knowledge (i.e. how to deal with the class imbalance problem), we concluded that such knowledge tends to most appropriately take a procedural or rule-like form. However, during early experimentation we quickly encountered several problems with attempting to use an OWL DL-based ontology for expressing procedural or rule-like knowledge:

- DL-based ontologies are declarative in nature.
- Using ontology query languages (i.e. SPARQL, RDQL) for emulating reasoning mechanisms was deemed unmanageable.

With respect to using ontology query languages for our reasoning needs, the shortcomings mainly resulted from the fact that, though query languages can be very useful for supporting the information retrieval process, such languages do not possess a syntax that is suitable for supporting formal reasoning paradigms (i.e. forward-chaining, backward-chaining, etc.). Nonetheless, the above problems were resolved by making use of an ontology comprising of declarative elements (i.e. Concepts, Properties and Individuals), rules and an external rule-based inference engine (JESS). The rules were implemented and integrated into an ontology using the proposed rule-language standard for the semantic web - SWRL [103].

4.4.1 Ontology Knowledge Elicitation

The process of eliciting detailed DM knowledge for our ontology consisted of the following 5 steps:

1. **Concept Taxonomy Elicitation** – Captures the high-level DM and CRISP-DM methodology concepts in the form of a taxonomic structure (from general to specific when possible).
2. **Properties Description** - Used to acquire case attributes as predicates which can eventually be used to express detailed knowledge in the form of rules.
3. **Individuals Definition** - Captures the individuals that shall later be used in the antecedent and consequent parts of rule expressions.
4. **Advice Annotations** – These capture the specific textual recommendations and heuristics that will be presented to the user as a result of the reasoning process.
5. **SWRL Rules Elicitation** – Used to define DM procedural knowledge by making use of the above previously defined elements (i.e. Properties, Individuals and Annotation Properties).

The first four steps are discussed in the following Section (4.4.1.1), while the last step - SWRL Rules Elicitation - is covered in Section 4.4.1.2. Our objective has been to restrict the elicitation of detailed DM knowledge to the key phases of the CRISP-DM methodology such as the Data Understanding, Data Preparation, Data Modeling and certain parts of the Evaluation phase using common DM algorithms. It is worth noting that though we have presented the above knowledge elicitation process in a systematic fashion, it has more likely resembled the iterative nature of the knowledge engineering process previously discussed in Section 3.6 (CommonKADS).

4.4.1.1 Concepts, Properties and Individuals Elicitation

The first step in eliciting DM knowledge consisted of capturing a taxonomy of concepts as illustrated by the solid oval shapes in Figure 22 and Figure 23. The next step involved the definition of a set of properties (analogous to the DM case attributes from Section 4.3.2) from which these could be used as predicates within rule expressions or procedural knowledge. Essentially, the definition of such properties provided a bridging mechanism for the transfer of CBR case attributes onto the DM ontology and their eventual use as “facts” within rule expressions during reasoning. For the moment, reasoning details have been deferred to Section 4.4.2.

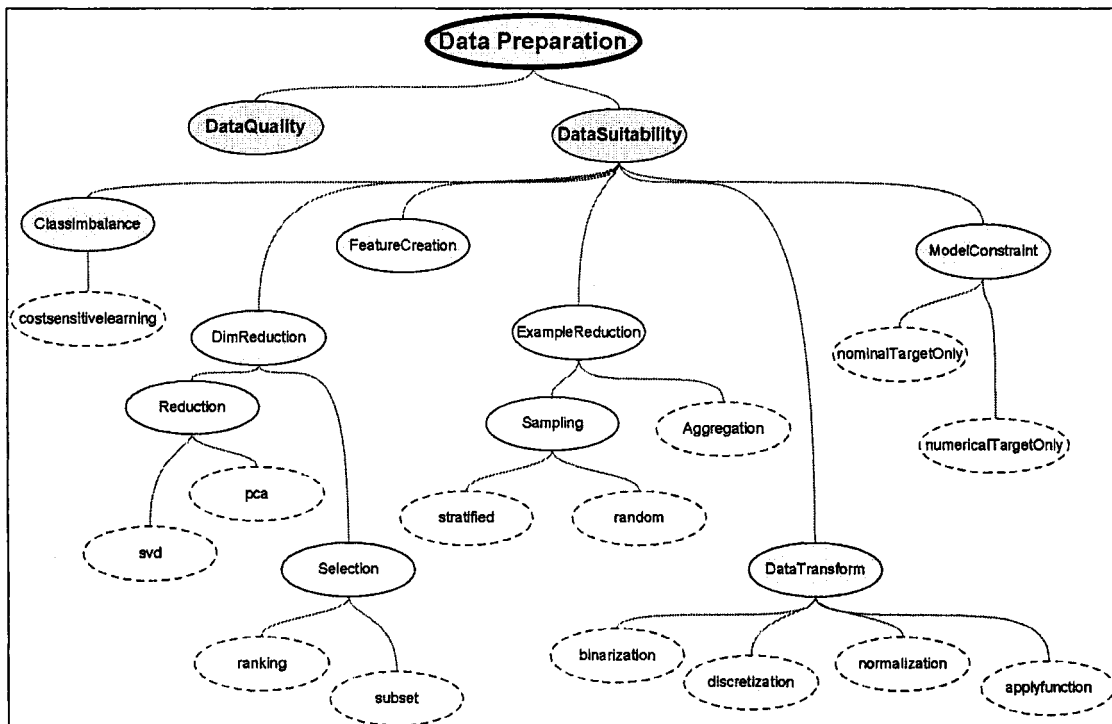


Figure 22 Data Preparation Knowledge Elicitation Taxonomy

Figure 22 illustrates concepts that are commonly associated with the various activities of the data preparation phase. For instance, the figure presents a taxonomic structure of concepts relating to data suitability issues such as class imbalance, dimensionality reduction, examples reductions and modeling constraints. Furthermore, the figure

demonstrates that the *ExampleReduction* concept defines a specialization or inheritance relationship with two sub-concepts such as the *Sampling* and *Aggregation* techniques, while the *Sampling* concept contains two specific instances or individuals - the *stratified* and *random* sampling techniques.

Since SWRL rules primarily operate on individuals, the third step consisted of the definition of individuals (or instances) that were to participate in the antecedent and consequent parts of the SWRL rule expressions (see next section for details). The last step (and most important from the user's perspective) consisted in defining annotation properties that contain textual recommendations and heuristics intended for the user. Essentially, these textual annotations are the by-product of the reasoning mechanism. These annotation properties are "grafted" to the various individuals illustrated by dashed ovals within Figure 22 and Figure 23. Section 5 shall be demonstrating actual recommendations (in the form of screen captures) as they are presented to a user during the operation of our intelligent DM assistant.

Though a full graphical description of our DM ontology is beyond the scope of this report, Figure 23 illustrates the taxonomy of supported DM algorithms or models. For example the figure presents high-level categories of machine learning algorithms or models such as tree-based, lazy learning or functional approaches (respectively represented by the *Tree*, *Lazy* and *Functional* concepts). The tree-based approach contains two specific individuals – the *ID3* and *J48* algorithms, while the functional-based approach contains a *Regression* concept (with Linear and Logistic individuals), and two additional individuals - the *RBFnetwork* and *SMO* algorithms. Essentially, the dashed oval shapes (individuals) represent the terms or components that may be used to form SWRL rules.

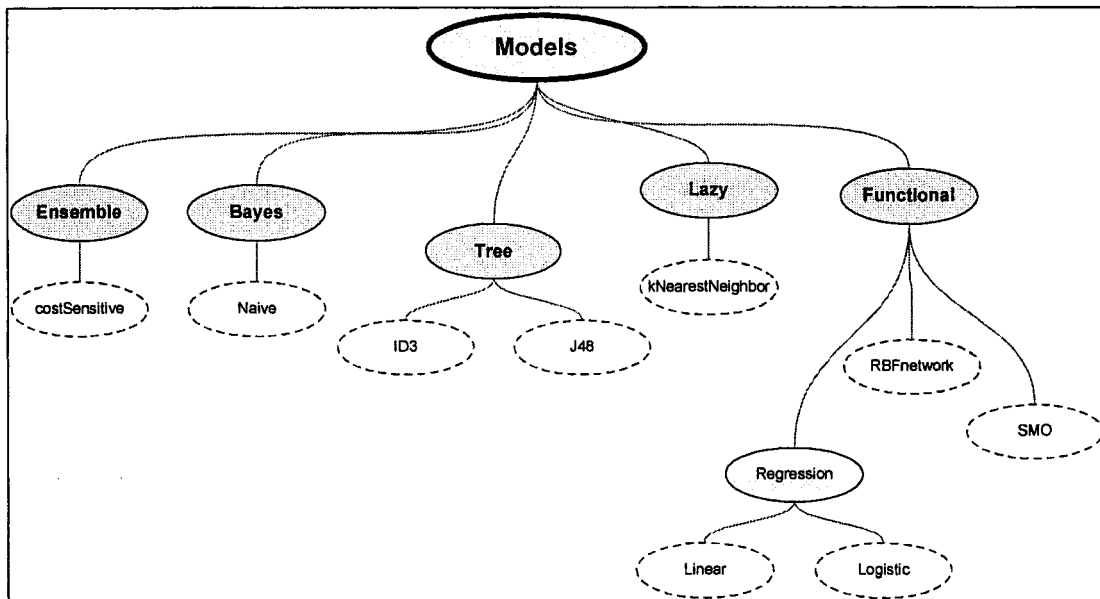


Figure 23 Data Mining Models Taxonomy

4.4.1.2 SWRL Rules Elicitation

We have elicited a set of rules for providing intelligent DM assistance using the *SWRL Rule Editor* plug-in of the Protégé ontology editor (as previously discussed in Section 3.5.5). The use of SWRL rules has provided two important benefits for the realization of our system:

- A convenient method for expressing domain knowledge as a set of antecedent-consequent pairs.
- Provide an integration mechanism for bridging knowledge from two disparate knowledge bases (CBR and ontology).

Due to space limitations the examples herein shall only be limited to the Data Preparation phase (some rules are illustrated as dashed lines in Figure 25). Hence, the following rules represent detailed knowledge that may be required for successfully performing the data preparation phase. The SWRL rule below asserts that if the problem case (pc) has an “example count” greater than 30000 and the dataset is of a

“transactional type”, the user should consider performing an aggregation operation over the dataset. An arbitrary “adaptation case” (ac) individual is used for holding advice values¹⁸.

```
Rule1 := NoExamples(pc, ?x1) ^ swrlb:greaterThan(?x1,
30000) ^ transactionData(pc, True) -> advice(ac,
aggregation)
```

Furthermore, *Rule2* below essentially expresses that if a binary class problem has its minority class represented by less than 15%, a class imbalance problem may be eminent:

```
Rule2 := numOfClasses(pc, 2) ^ minorityClass(pc, ?x1) ^
swrlb:lessThan(?x1, 0.15) -> advice(ac, classImbalance)
```

Hence, the *classImbalance* individual provides advice by offering a cost-sensitive learning algorithm to attempt to improve overall model performance. In addition, the following rule asserts that if the quantity of attributes is greater than 20 (but less than 50 as PCA can be computationally prohibitive) and the “symbolic attributes ratio” is zero (only numerical values) then the system would recommend specifically using the PCA dimensionality reduction technique:

```
Rule3 := noAttributes(pc, ?x1) ^ swrlb:greaterThan(?x1, 20)
^ swrlb:lessThan(?x1, 50) ^ ratioSymbAttributes(pc, 0) ->
advice(ac, PCA)
```

See Appendix D for an unabridged list of all the SWRL rules that were implemented within our system. In summary, we have implemented an OWL-DL ontology of approximately 97 concepts, 58 properties, 63 individuals, 68 rules and 42 annotation properties (i.e. 30 recommendations and 12 heuristics).

¹⁸ The significance of the problem case (pc) and advice case (ac) are explained in the next section.

4.4.2 The Rule-Based Inference Component

Detailed DM knowledge is made available to the user via an event-driven reasoning cycle. Our reasoning cycle consists of the following 5 phases:

1. **Importing Facts** – Acquire case attribute information from the user interface and populate these into the ontology.
2. **Exporting Knowledge and Rules** – Exporting pertinent knowledge and rules to the rule-based reasoner (JESS).
3. **Reasoning** – Perform a forward-chaining reasoning procedure to infer new facts (i.e. recommendations).
4. **Assertion of New Facts** – Assert new facts into ontology (some are intermediate facts and others are actual recommendations).
5. **Advice Forwarding** – Forward the newly asserted facts (advice) to the user interface.

In brief, during operation of our DM assistant the user will perform attribute changes to the information grid or DM case facts. These facts must be continuously imported into the ontology via an event-driven mechanism (using the Protégé API). Precisely, these facts are “grafted” (a relation is formed) onto an specific problem case individual (indicated by PC in Figure 24). From this, all pertinent knowledge is exported to the JESS engine (via the SWRL-Jess-Bridge Java API). When appropriate, the JESS engine performs forward-chaining inference on these (rules are “fired”) and asserts new facts. Subsequently, new facts are asserted into the ontology using an advice case individual (indicated by AC in Figure 24). Last, but not least, these newly asserted facts are forwarded to the user interface and provide recommendations and heuristics to a user on how to perform a correct case adaptation. This reasoning cycle is graphically illustrated in Figure 24. The arrows numbered 1 to 5 correspond to the above-

mentioned reasoning phases, while the arrow numbered 0 represents the method initially used for eliciting or defining the rule set into the ontology.

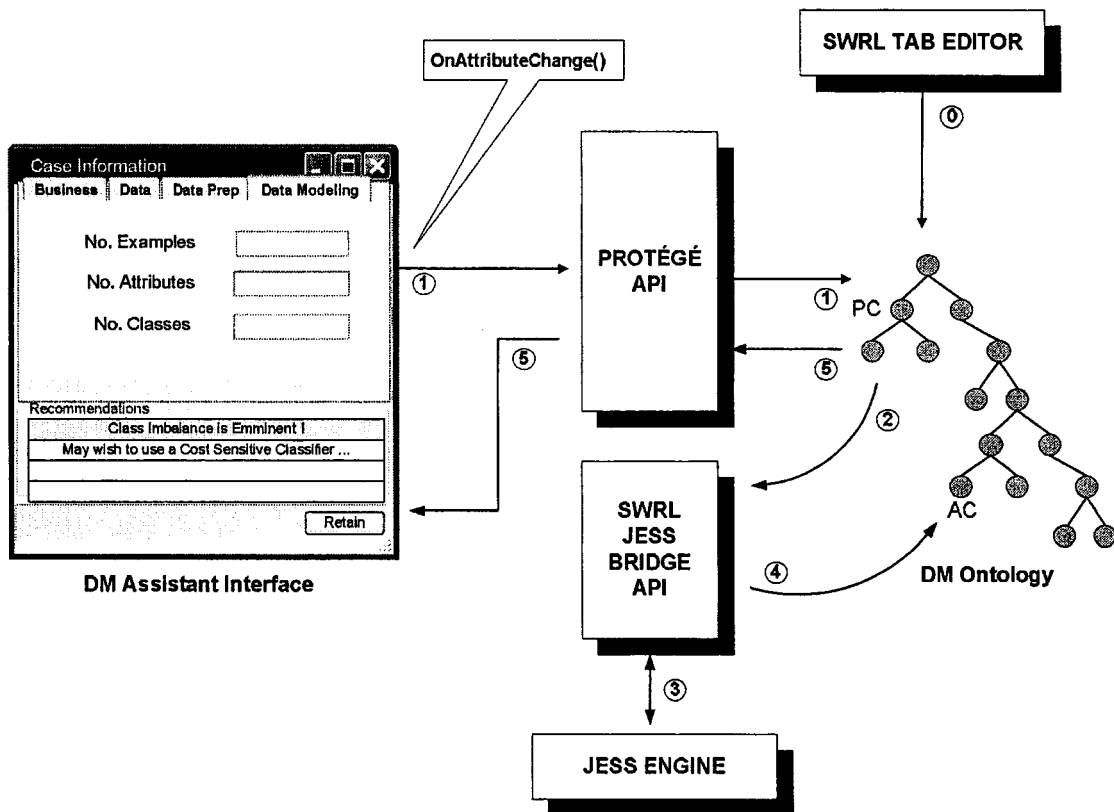


Figure 24 Conceptual View of the Reasoning Cycle

4.4.3 Ontology-Guided Intelligent DM Assistance

This section addresses how the above-mentioned system components (DM ontology and reasoning cycle) are synergistically combined to provide a novice data miner with ontology-guided DM assistance. In order to facilitate our discussion, Figure 25 essentially provides a conceptual view of the principle components; sample DM case attributes are represented by the *DM Assistance Information Grid*, an ontology segment represents some detailed DM knowledge and several SWRL rules are abstracted as dashed lines. Although the CBR paradigm provides the benefit of retrieving similar cases, the required solution part is rarely an exact match to the current DM problem

being attempted. Hence, after the retrieval and reuse CBR phases are completed, the user is faced with the grand challenge of examining the chosen basis cases' contents and revising certain attributes in order to retrofit the case to reflect the state of the current DM problem.

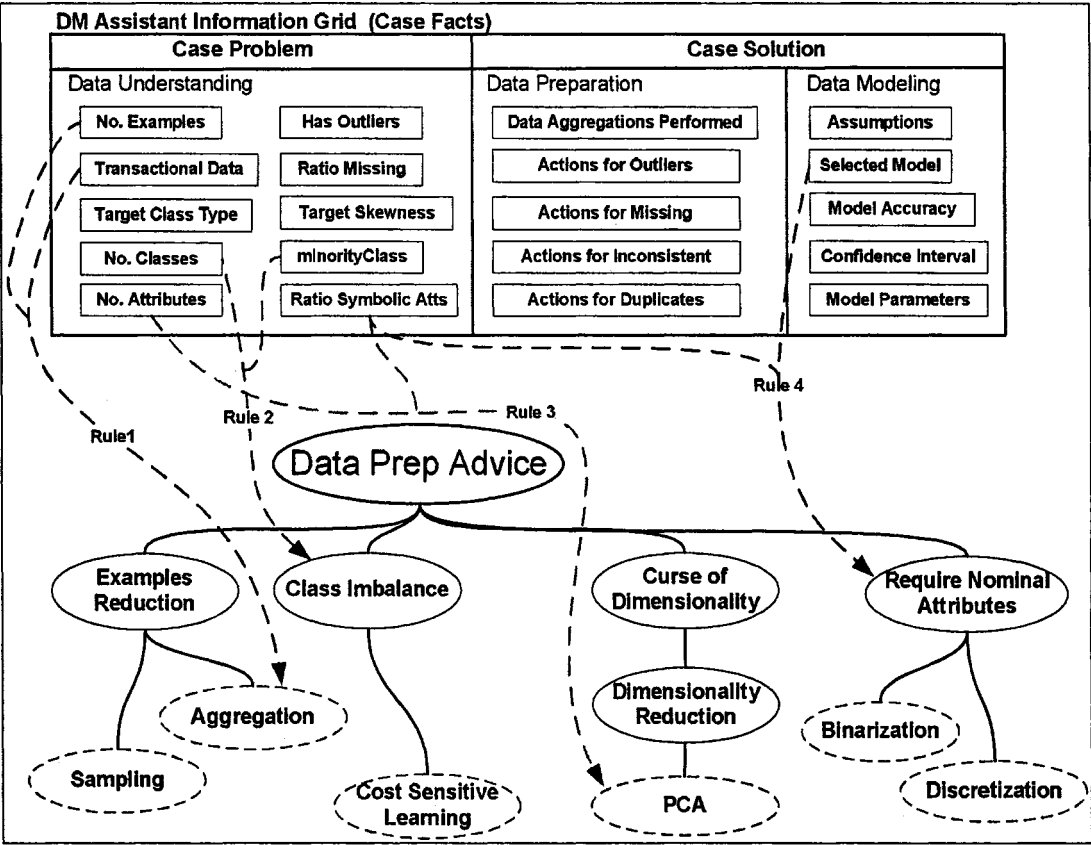


Figure 25 A Conceptual View of CBR and Ontology Synergy Using SWRL Rules

The concepts illustrated in the ontology segment of Figure 25 (starting from the root *Data Prep Advice* concept), represent important potential data mining problems that can have a significant impact on the final quality of a generated model. For instance, some algorithms can perform poorly if the quantity of examples becomes large (*Examples Reduction* in Figure 25), while other machine learning algorithms can be significantly affected by too many attributes (*Curse of dimensionality* in Figure 25). An experienced data miner can fairly easily mitigate these problems by applying a supplementary procedure (i.e. aggregation, a cost-sensitive learning method, etc in Figure 25.). Specific advice for a given problem is represented by ontology individuals as indicated by dashed ovals¹⁹. The system is essentially data driven and employs a forward-chaining rule-based inference engine (JESS). A user basically interacts with the *DM Assistant Interface* (the *DM Assistant Information Grid* in Figure 25) by entering or modifying a series of DM case attributes (i.e. facts). The abbreviated *DM Assistant Information Grid* represents the state of the “working memory” of the system. As the user changes the state of the working memory, the SWRL rules come into play to provide advice and heuristics (i.e. which facts to modify and what values to enter when possible) in the form of textual messages. The purpose of the messages is to actively assist and empower the user to provide acceptable fact values during the case adaptation process.

It is worth noting that we originally provided automated fact responses (automatically changing the state of the DM case for the user as rules fired), however it was quickly ascertained that such behavior posed several problems. The most notable was that certain DM case changes could occur unnoticed by the user and promote further confusion. In addition, we believe that it is best for the user to make the actual attribute changes and actively learn during the process of case adaptation.

¹⁹ Actually, it is the annotation properties that are associated with these individuals that contain the actual text-based recommendation.

4.4.3.1 The Initial Bootstrap Advice

Under ideal circumstances, the state of the initial working memory should be adequately specified from automatically provided facts (i.e. ratio of missing values and other data characteristics provided by data characterization module, etc.) to allow the firing of certain rules to move the DM process forward. Nevertheless, there are circumstances when user input is required (i.e. identification of outlier or incomplete values within the problem set). When such facts are required directly from the user, initial textual messages (bootstrap advice) are given to the user, explaining how to acquire the missing information. This approach is analogous to traditional AI interview or conversational techniques used for soliciting tacit information from the user.

4.4.3.2 Terminological Definitions

As an aside, the DM ontology also provides the user with basic definitions for all the vocabulary terms used within the *DM Assistant Interface*. Though this information does not involve any reasoning per se, it does provide the user with a lexical or dictionary-like representation from which to learn the meaning of basic DM terms. These terminological definitions were made available via the use of SPARQL queries (implemented using the Jena API) on annotation properties within our ontology. For example, the following is an example SPARQL query that was used to retrieve a text-based definition for the Data Preparation concept:

```
SELECT ?comm WHERE { :dataPreparation :description ?comm }
```

The above query will acquire the text string (?comm) associated with the *dataPreparation* individual that is specified by the *description* property (or predicate).

The actual textual definition is:

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Tasks include table, record and attribute selection as well as transformation and cleaning of data for modeling tools.

4.4.3.3 Recommendations and Heuristics

Although we shall be using the general term "advice" to represent any assistance provided by the system (i.e. text message), we do make a clear distinction between a recommendation and a heuristic (both are sub-types of the term advice). A recommendation is a more formal type of advice (assertion), while a heuristic should be interpreted less formally by a user (i.e. rule-of-thumb). Since it is a fact that the Naïve Baye's algorithm requires a nominal target, the following is an example of a recommendation rule:

```
Rule4 := selectedModel(pc, "naiveBayes") -> DPadvice(pc,  
    requirenomtarget)
```

On the other hand, the previously defined rules (*Rule1*, *Rule2* and *Rule3*) in Section 4.4.1.2 are examples of heuristic rules since these are not theoretically founded (or a fact), but are derived empirically as guidelines during DM activities. For instance, the defined minority class limit of 15% for *Rule2* could as well be 10% or 20% since there is no theoretical proof for such a fixed value when dealing with a class imbalance problem [45].

4.4.3.4 Scope of Detailed DM Knowledge

Since the area of data mining is a highly knowledge-rich environment (i.e. data cleansing, feature transformation, algorithms, parameters, evaluation, etc.), it is impossible to foresee capturing all the DM knowledge that is required to support users under all conceivable circumstances. Hence, our current prototype's detailed ontology knowledge (though not exhaustive) is currently constrained to the following:

- Support the data preparation phase for handling common data quality and model input requirements.
- Support for common classification models (i.e. linear/logistic regression, naïve bayes, most decision trees, support vector machines).

- Common data modeling issues (i.e. class imbalance, curse of dimensionality, basic model over-fitting avoidance).
- General knowledge for model evaluation (i.e. P-values, cross-validation, ROC curves).
- Specific tool dependent knowledge is only available for the Weka environment.
- More advanced topics such as meta-learning, feature selection, massive datasets, model comparison methods and intricate classifier parameter details are not yet covered.

The detailed DM knowledge (primarily in the form of SWRL rules) was mainly elicited from introductory data mining texts ([104], [118]), the Weka mailing list [115], scientific articles (for example, [29] and [55]) and our own DM experiences. Realistically, our objective has been to elicit a “first-pass” to capture common DM knowledge and subsequently evolve our ontology iteratively as the needs arise (i.e. to handle specialized and exceptional DM process conditions).

4.4.3.5 A Note on Rule Opacity

It is worth noting that the SWRL rules could have been implemented purely using propositional rules (without using ontology concepts and individuals). Nevertheless, we believe that the formal capture and representation of detailed DM knowledge within an ontology provides some important benefits:

- It provides a more explicit form of knowledge representation that is more amenable to human interpretation.
- Unlike traditional rule-bases where the relationships between the rules tend to be opaque, the explicit representation of linguistic variables as formal ontology concepts facilitates rule-set reuse and maintenance.

- Knowledge management efforts can be performed in several independent stages (and possibly by independent domain experts). For instance, declarative DM knowledge can first be elicited, and subsequently another domain expert can make use of this knowledge to craft a set of SWRL rules for expressing procedural DM knowledge.

4.5 Intelligent DM Assistant System Overview

Having surveyed the knowledge elicitation methods and inference mechanisms (i.e. case-based and rule-based) used for both the CBR and ontology sub-systems, we finally proceed to a brief system overview to show how all the parts of the system work synergistically to offer intelligent DM assistance.

As illustrated in Figure 26, our hybrid DM assistant consists primarily of six major components: a *DM Case Base*, a *DM Ontology*, a *Case Reasoner*, *Rule Reasoner*, a *DL Reasoner*, and a *DM Assistant Interface*.

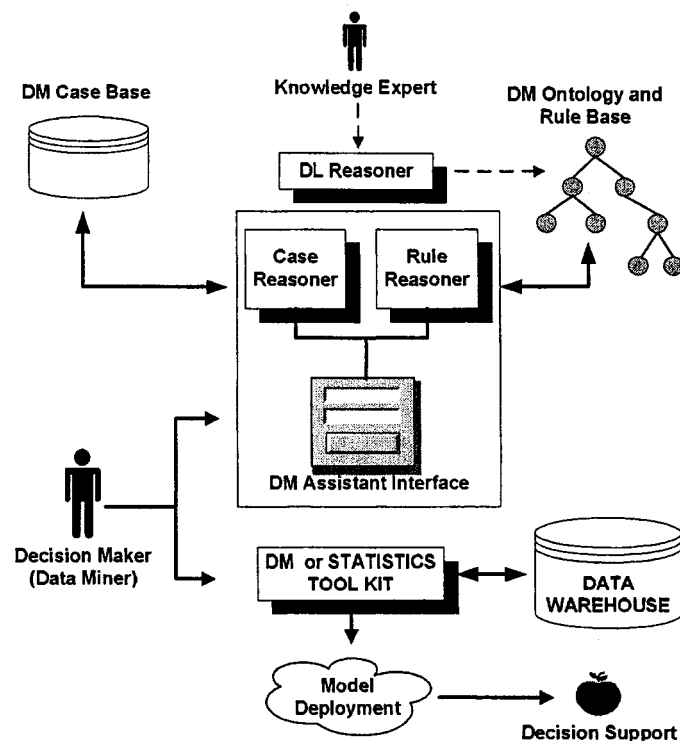


Figure 26 Intelligent Data Mining Assistant System Overview (source [21])

The best way for a user to profit from our intelligent DM assistant is to use it in parallel with a DM toolkit. Essentially, the user (*Data Miner* in Figure 26) begins a DM assistance session. The system will perform an initial problem characterization of the user's current problem and guide the user to making use of knowledge from a previously resolved 'similar' DM basis case. From this, as the user proceeds through the DM process, the DM assistant shall provide advice on how to use the chosen DM toolkit. From other parts, the system expects the chosen DM problem dataset to reside within a data warehouse or RDBMS. Last, hopefully with advice from the system, the user shall produce a suitable model which can be deployed and used for its intended business purpose (illustrated by apple in Figure 26).

Interestingly, the CBR and DM ontology subsystems have well defined knowledge representation roles. The DM ontology defines and manages high-level concepts (i.e. tasks, activity types, algorithms, etc.) while the CBR holds detailed case information (i.e. data preparation steps, model parameters, etc.). The CBR system is capable of learning useful "DM problems to DM solutions" knowledge while the DM ontology provides additional assistance (complements where the CBR lacks knowledge) to a user during the various phases of the DM process in the form of textual recommendations and heuristics.

4.5.1 Implementation Issues

Figure 27 illustrates the organizational topology of the key software components (both used third-party and implemented components) that make up the intelligent DM assistant. Essentially, the server computer houses various subsystems such as the *Case Base*, *DL Reasoner*, *DM Ontology*, the *JESS Rule-Based Reasoner* and *Business Data*. Notably, the majority of the core processing components that we implemented were deployed using the Tomcat Web Container [106]. These processing components (i.e. *CBR core*, *DM Assistant* and *Ontology Processing Core*) were implemented as library modules using a combination of the Jython [51] and Java [46] programming languages, while the *DM Assistant* was implemented as Servlet technology that uses the modules [94]. The event-driven mechanism was implemented using Javascript technology [47] and other APIs previously mentioned in Section 4.4.2 (i.e. Protégé,

JESS and SWRL-Jess-Bridge APIs). The *Case Base* and *Business Data* repositories were implemented using the MySQL relational database [67] (though they were also successfully deployed onto the Oracle database server [73]). The most frequently used *Data Mining Toolkits* were the Oracle Data Miner [74] and the Weka DM toolkit [115]. Last, but not least, the Pellet DL reasoning API [77] was at times used for performing consistency and integrity verifications on our DM ontology during knowledge elicitation efforts.

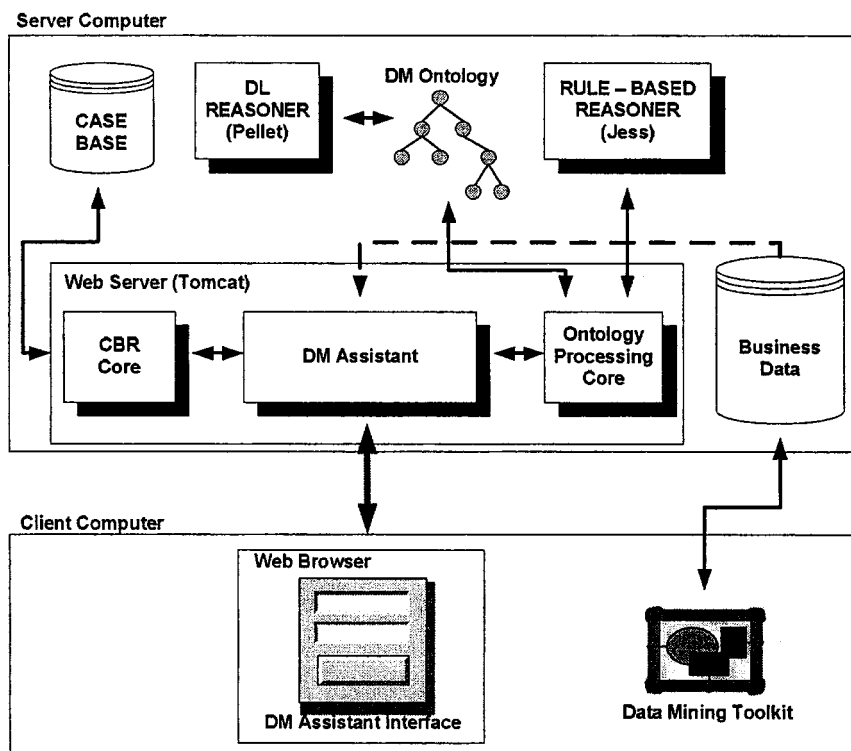


Figure 27 A Deployment Diagram of the DM Assistant

Having been introduced to the key conceptual, architectural and technological components that have fostered the realization of our intelligent data mining assistant, the following chapter presents several examples where our DM assistant was applied, as well as the verification method that was used for performing a qualitative evaluation of the system.

CHAPTER 5

Tests, Results and Validation

This chapter first presents a quick tour of the intelligent data mining assistant. We then examine and compare how the data mining assistant provides recommendations for several DM problems with those offered by a human data mining professional. Last, using the results obtained from these DM problems, we provide a brief system evaluation of our intelligent data mining assistant.

5.1 A Quick Tour

Figure 28 below presents the main web interface of the intelligent data mining assistant. The interface primarily consists of 4 buttons. The *START* button is used to begin a data mining assistance session. Typically, a user will work concurrently with a data mining toolkit (i.e. Oracle Data Miner or Weka) and the data mining assistant.

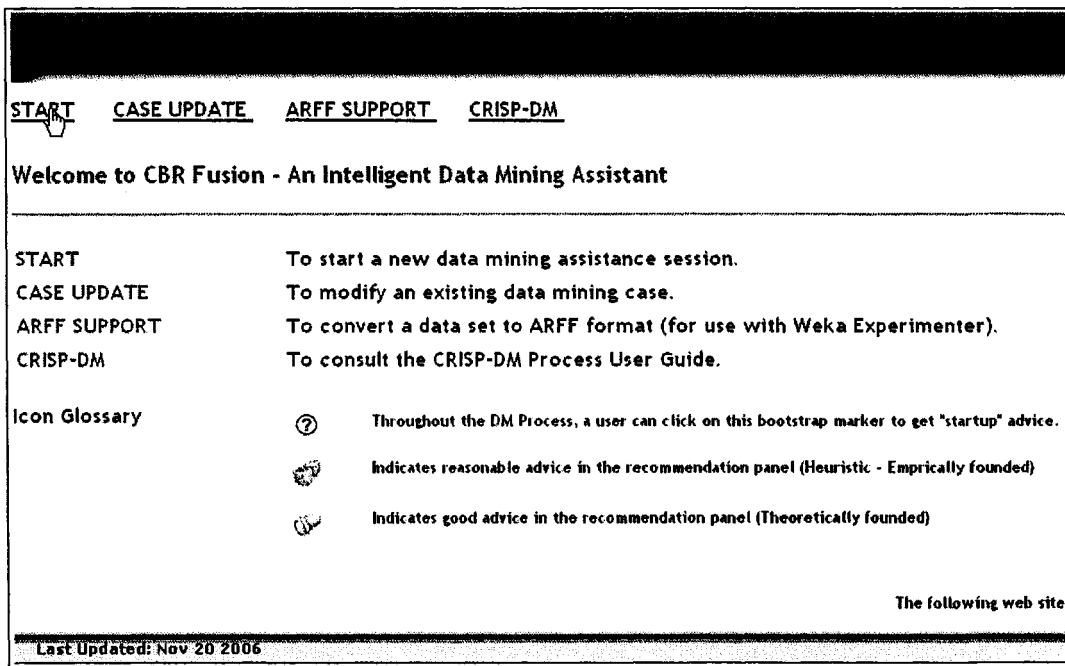


Figure 28 Intelligent Data Mining Assistant Main Interface

The *CASE UPDATE* button is used to perform minor updates to previously stored DM cases. This option is particularly useful post-deployment to make minor modifications to a DM case once a model is used in a real business setting. The *ARFF SUPPORT* button is used to provide a user with ARFF file conversion support for the *Experimenter* component of the Weka DM suite. Last, the *CRISP-DM* button provides the user with direct access to the CRISP-DM User Guide documentation. This principal page also provides an icon glossary and explains the meaning of graphical symbols commonly encountered during a DM session. Once the user has pressed the *START* button, a data source specification form appears as illustrated in Figure 29. In this step, the user specifies the data source of interest from which subsequent data preparation and modeling activities will be applied. Hence, this form contains fields for specifying the database, user and password details, the table (or relation) name, a list of attributes and a target class of interest. By default, the DM assistant accesses data sources from a locally installed MySQL database server. For example, in Figure 29 the user has specified a data source from the *uqtr* database, the *uqtr20043* table and is interested in working with 2 attributes (i.e. *cd_pgm*, *crd_reussis*) and the *etat_pgm* target class.

HOME HELP	
Step 1: Data Source Specification ... and Characterization	
Database	<input type="text" value="uqtr"/>
User	<input type="text" value="uqtr"/>
Password	<input type="password" value=""/>
Table Name	<input type="text" value="uqtr20043"/>
Attribute List	<input type="text" value="cd_pgm, crd_reussis"/>
Target Class	<input type="text" value="etat_pgm"/>
Server Type	<input type="text" value="MySQL"/>
<input type="button" value="Characterize Data Set"/> <input type="button" value="Clear"/>	
version: FA1	

Figure 29 Data Source Specification and Characterization Step

Having specified a data source to work with, the DM assistant automatically computes various data characteristics (see Section 4.3.3 for details) and presents the following problem characterization form as illustrated in Figure 30. In order to completely characterize the initial DM problem, the user must complete 4 remaining problem characteristics as illustrated in the form below. Since these attributes (those having associated question mark symbols) are currently not automatically determined by the system, the user can consult the question icon to obtain “bootstrap advice” on how to select appropriate values for these particular DM case attributes. For instance, a user may select the “follow-up” *Business Area*, the “classification” *DM activity* and answer “no” for the presence of both *outlier* and *inconsistent* values. The small popup window in Figure 30 illustrates the bootstrap advice offered for selecting the *Business Area* attribute of a given DM problem. Having completed the problem characterization step, the user will typically press the *Retrieve Similar Cases* button in order to acquire a set of previously resolved similar DM cases for the given problem characterization.

Step 2: Problem Characterization ... and Retrieval

Business Area	SELECT ?	?	<div style="border: 1px solid black; padding: 5px;"> <p>Definition:</p> <p>The business area attribute is used to discriminate amongst a finite possible set of data mining application areas within a given domain (i.e. engineering, finance, marketing, etc.). In the current context of applying data mining within a University setting, the following three typical business areas are available: (1) admission - this sub-area should be selected for analyzing data that is gathered from the admission process; (2) retention - this sub-area should be selected for carrying out data mining activities on student data corpus that is collected during regular operations; (3) follow-up - this sub-area should be selected for analyzing data that is collected from various surveys (i.e. ICOPE).</p> <p style="text-align: right;">Terminé</p> </div>
DM Activity	classification	?	
Number of Examples	1000		
Number of Attributes	2		
Number of Class Values	3		
Ratio of Symbolic Attributes	0.5		
Mean Skewness	-0.0297		
Mean Kurtosis	1.7355		
Normalized Class Entropy	0.536		
Maximum Mutual Information	0.4873		
Target Class Type	nominal		
Ratio of Duplicates Records	0.0		
Has Outliers Values	SELECT ?	?	
Ratio of Missing Values	0.0		
Has Inconsistent Values	SELECT ?	?	

OR

version: PAI
LQTA

Figure 30 Bootstrap Information and the Retrieval Step

Having provided problem characterization details, the CBR retrieval process is carried out and provides the user with a set of similar, previously resolved DM cases that are available within the case base as illustrated in Figure 31. This table presents the 3 most similar cases (sorted in descending order of similarity). Subsequently, a user will examine the GUM and GSM columns in order to assess which DM case is most suitable to use (and subsequently adapt) in order to resolve the current DM problem at hand. For example, as illustrated in Figure 31, although case no. 4 has scored a higher GSM rating, the user would be better off selecting **case No. 1** as a basis case, since it offers the best trade-off between GSM and GUM measures. As previously explained, the GUM value provides a measure of the “usefulness” or potential for a given case to aid the non-expert during the case adaptation phase. Moreover, the *DU*, *DP* and *DM* columns in the figure respectively indicate the utility values scored at each phase (i.e. Data Understanding, Data Preparation and Data Modeling).

Step 3: Selecting a Basis Case for the Current DM Problem						
Case ID	Name	DU	DP	DM	GUM	GSM
4	Predicting Student Status Change Graduation Year	0.07	0.17	0.1	0.334	0.96
1	Predicting Student Program Status	0.2	0.5	0.2	0.9	0.878
3	Predicting Reasons for Student Program Status Change	0.1	0.4	0.2	0.7	0.838

Legend	
DU	Data Understanding Utility Score
DP	Data Preparation Utility Score
DM	Data Modeling Utility Score
GUM	Global Utility Measure
GSM	Global Similarity Measure

Figure 31 Selecting a Suitable DM Basis Case

Having selected a basis case to work with, the system will present the following DM case details form as shown in Figure 32. The DM case details are divided using tabs into the 5 principal phases of the CRISP-DM process (i.e. Business Understanding, Data, Preparation, Data Modeling and Evaluation). Although more advanced DM users may benefit from examining the specific DM case details in order to perform an initial assessment, it is recommended for novice data miners to simply confirm the DM basis case selection by pressing the *REUSE* button. It is worth mentioning that a user may at any time go back to the initial case retrieval form (Figure 30) to repeat the process should certain values need to be modified or corrected.

Business Understanding	Data	Preparation	Modeling	Evaluation
Step 4: Reviewing the Details of a Similar Case				
Business Objectives				
Case Name	Predicting Student Program Status			
Business Problem	Implement a predictive model for assessing student program status based on profile information.			
Business Area	follow-up			
Success Criteria	Deploy the classifier within the organization and obtain an 80% success rate.			
Situation Assessment				
Potential Data Source	ICOPE 1996			
Data Mining Goals				
DM Problem	Implement a classifier using ETAT_PGM as the target class.			
Expected Model Accuracy (min.)	75			
DM Activity Type	classification			
Project Planning				
Tools Assessment	Weka			
				<input type="button" value="Reuse"/>

Figure 32 Reviewing the Basis Case and the Reuse Step

After having committed a basis case for reuse, a case revision and recommendations interface panel as illustrated in Figure 33 is presented to the user. Essentially, this case revision and retain form shall serve as the principal dynamic interface from which the user will receive recommendations and heuristics in order to resolve the current DM problem at hand.

When the chosen similar basis case is loaded into the interface (along with the current DM problem characteristics), the rule-based inference engine operates (reasons) on the case attribute values (or facts). As a result, several recommendations and heuristics appear in the right hand side panel as indicated in Figure 33. For example, the previous case did not require a handling of inconsistent values (*Inconsistent Handling* is set to *non-applicable*), but since inconsistent values have been defined for the current DM problem, the system is recommending that the user provide a different attribute value. In addition, an additional recommendation is provided since duplicate values are present and the *Duplicates Handling* attribute is set to *eliminate*.

Business		Data		Preparation		Modeling		Evaluation	
Step 5: Revise and Retain the Case									
Data Selection									
Feature Reduction Method	SubsetSelection								
Feature Reduction Details	removed sexe attribute as is was not significant								
Example Reduction Method	non-applicable								
Data Cleaning									
Duplicates Handling	eliminate								
Duplicates Details	eliminated since there were only 10 examples.								
Outliers Handling	estimateUsingMeanMode								
Outliers Details	COTE_RENDEMENT had 276 examples with value 0.								
Missings Handling	non-applicable								
Missings Details	no missing values.								
Inconsistents Handling	non-applicable								
Inconsistents Details	no inconsistent values.								
Data Construction									
Feature Creation Method	non-applicable								
Feature Creation Details	not required.								
Data Integration									
Data Aggregation Performed	no								
Aggregation Details	this was not transactional data.								
Data Formatting									
Data Transformation	non-applicable								
Transformation Details	target class was nominal as expected.								
Revert					Retain				

Recommendation R

Inconsistent values have been detected (HAS INCONSISTENT VALUES is yes). An inconsistent value can occur when an attribute value is not one of the expected or permissible values (i.e. an attribute containing a misspelled country or invalid postal code). Typically, inconsistent values are caused by data collection errors (i.e. inadvertently transposing the digits of a postal code field). Inconsistent values may be the cause of either an incomplete value (i.e. missing digits for a product code) or an invalid value (i.e. an negative value for the height of a person). Regardless of the cause of an inconsistent value, it is important to detect, and when possible correct or estimate such values.

When dealing with data quality issues such as duplicates, missing values, outlier values and inconsistent values, one should consider elimination as the last option for handling these. With more careful data analysis, it may be possible to correct or at least estimate these values. This can avoid losing potentially valuable data during the modeling (learning) process. It is advisable to reconsider your handling options where an eliminate is currently specified.

Figure 33 Recommendations and the Case Adaptation Step

It is worth mentioning that context sensitive textual information is available throughout the entire process in order to guide the user to answer the various DM case attributes as accurately as possible. For example, Figure 34 illustrates a textual message appearing in the form of a tool-tip which provides the user with a basic definition or explanation of the *Business Objectives* concept as it applies to the CRISP-DM methodology.

Business		Data		Preparation		Modeling		Evaluation	
Step 5: Revise and Retain the Case									
Business Objectives									
<p>The first objective of the analyst is to thoroughly understand, from a business perspective, what the client really wants to accomplish. Often the client has many competing objectives and constraints that must be properly balanced. The analyst's goal is to uncover important factors at the beginning of the project that can influence the final outcome. A likely consequence of neglecting this step would be to expend a great deal of producing the correct answers to the wrong questions.</p>									
Business Area	rentention								
Success Criteria	Deploy the classifier within the organization and obtain an 80% success rate.								

Figure 34 Context Sensitive Support for CRISP-DM Terminology

5.2 Assisted Data Mining Problems

This section presents four data mining examples that attempt to demonstrate the assistance capabilities of our system. For the sake of brevity, the following examples have excluded possible recommendations provided by our assistant for the *Business Understanding* and *Process Evaluation* phases. The first example is presented in a detailed manner with screen captures in order to give the reader a true feel for how the DM assistant operates, however subsequent examples shall be summarized using a tabular format. The second and third examples are specialized examples that deal with commonly encountered situations such as a class-imbalance problem and a need for applying a specific feature reduction technique. The last example is presented in the form of a regression activity.

5.2.1 A Classification Example

This example was drawn from a student survey compiled in 2004. The dataset consists of approximately 3000 examples, 5 attributes and a nominal target class. The DM objective consists in producing a classifier for predicting student program status (*etat_pgm*) such as active, inactive and interrupted. The problem makes use of 4 dependent attributes such as the particular program (*cd_pgm*), grade point average (*moy_cum_pgm*), college performance rating (*cote_rendement*) and number of credits completed in the program so far (*crd_reussis*). Having specified data source details to the DM assistant, Figure 35 illustrates the obtained similar cases from our case base. It is worth noting that this DM exercise also represents the first DM problem resolved using our case base which initially contained only 3 seed cases. Hence, under such particular conditions (all retrieved cases score maximum GUM values), the obvious choice for the basis case for this DM example is simply the case which scored the highest GSM value (**case no. 3**).

Step 3: Selecting a Basis Case for the Current DM Problem						
Case ID	Name	DU	DP	DM	GUM	GSM
3	Predicting Reasons for Student Program Status Change	0.2	0.5	0.3	1.0	0.901
1	Predicting Student Program Status	0.2	0.5	0.3	1.0	0.874
2	Predicting Student GPA	0.2	0.5	0.3	1.0	0.511

Figure 35 Selecting a Basis Case for a Classification Example

Figure 36 illustrates the basis case details for the data preparation phase. Although existing case attribute values may appear less useful (without explicit recommendation messages), these can provide useful knowledge on how to carry out a given problem. For example, the *Data Selection* section recommends using a *FeatureRanking* technique for reducing irrelevant attributes from the dataset. In actual fact, we applied this technique and managed to remove an irrelevant attribute from our dataset (i.e. *crd_reussis*) and obtain a better model performance.

Step 5: Revise and Retain the Case	
Data Selection	
Feature Reduction Method	FeatureRanking
Feature Reduction Details	removed ANSES_DEB_PGM, HOY_CUM_CRD, HOY_CUM_POND
Example Reduction Method	non-applicable
Data Cleaning	
Duplicates Handling	eliminate
Duplicates Details	eliminated 20% of duplicates (resulted from primary key rer
Outliers Handling	eliminate
Outliers Details	Only 1 outlier value of GPA = -2.5
Missings Handling	estimateUsingMeanMode
Missings Details	5 missing values (nulls) for ETAT_PGM. replaced with mod
Inconsistents Handling	non-applicable
Inconsistents Details	no inconsistent values.
Data Construction	
Feature Creation Method	non-applicable
Feature Creation Details	not required.
Data Integration	
Data Aggregation Performed	no
Aggregation Details	this was not transactional data.
Data Formatting	
Data Transformation	non-applicable
Transformation Details	target class was nominal as expected.

Figure 36 Existing Case Recommendations for the Data Preparation Phase

Second, by examining the *Handling Missing* values attribute of the basis case, we were inspired to use the same technique for estimating our missing *etat_pgm* attribute by its mode (i.e. *estimateUsingMeanMode*). Third, although not as strikingly obvious, by examining *Outlier Handling*, *Outlier Details* and a recommendation as illustrated in Figure 37, we decided not to remove our outliers (as was done for case no. 3 since very few outliers were present) but to estimate these using the mean statistical operator (*estimateUsingMeanMode*).

Recommendation		R
<p>When dealing with data quality issues such as duplicates, missing values, outlier values and inconsistent values, one should consider elimination as the last option for handling these. With more careful data analysis, it may be possible to correct or at least estimate these values. This can avoid losing potentially valuable data during the modeling (learning) process. It is advisable to reconsider your handling options where an eliminate is currently specified.</p>		

Figure 37 Ontology-Driven Recommendation for the Data Preparation Phase

Having handled the data preparation phase, we now proceed to the data modeling phase of our DM problem. Figure 38 illustrates various data modeling case attribute information such as the selected model, testing options and model assessment results.

Step 5: Revise and Retain the Case	
Model Selection	
Modeling Assumptions	NominalTargetClass
Selected Model	naiveBayes
Test Design	
Train and Test Options	10 Fold Cross Validation
Model Design	
Model Parameters	Debug: False, UseKernelEstimation: False, UseSupervisedDiscretization: False
Other Parameters	non-applicable.
Model Assessment	
Estimated Model Accuracy	76.9 percent
Confidence Interval Limits	76.2, 77.5

Figure 38 Existing Case Recommendations for Data Modeling Phase

At this stage of the DM process, we applied the recommended model (*naiveBayes*) as well as a host of other models (i.e. *ID3*, *J48*, *RBFnetwork* and *SMO*) in search of the best model performance for our given problem. Nevertheless, the recommendation for using *naiveBayes* provided the best *Estimated Model Accuracy*. In addition, since the chosen model was the same as the basis case, we also benefited from using the same *Model Parameters*. The final estimated model accuracy was noted in our resolved DM case as 84.3 %. Furthermore, since our DM assistant detected that a classification problem was being resolved (rather than a regression problem or a class-imbalance problem), the system recommended that the *Estimated Model Accuracy* be used for evaluation as indicated in Figure 39 (see flashlight icon).



Recommendation	R
<p>The 10 fold cross validation divides the available dataset into 10 disjoint sets and uses 9 sets for training and the remaining set for testing (error rate estimation). This approach is the most popular method used in practice and is efficient for relatively small datasets (greater than 250 samples). Since the current data set is greater than 250 examples, the 10 fold cross validation is recommended, over the bootstrap method.</p> <p>Since the chosen model requires a nominal attribute, it is advisable to use the Estimated Model Accuracy (or error rate) as an indicator for model performance.</p>	 

Figure 39 Ontology-driven Recommendations for Data Modeling Phase

Last, but not least, we also followed the recommendation of using the *10 Fold Cross Validation* technique for assessing our model's performance. Actually, attempting to modify the *Training and Testing Options* to the *Bootstrap* method yielded the heuristic indicated in Figure 39 (see dice icon). Table 6 summarizes the recommendations that were provided by the system during the DM activity.

Table 6 Summary of Recommendations for a Classification Example

DM Phase	Case Attribute	System Recommendation	Human Expert Recommendation
DP ²⁰	Feature Reduction Method	Use Feature Ranking	Use Feature Ranking
	Outlier Values Handling	Remove outliers as per previous, but recommendation to carefully evaluate	Estimate with mean ❶
	Missing Values Handling	Estimate with mode	Estimate with mode
DM	Model Selection	Use Naïve Bayes	Use Naïve Bayes
	Train and Test Options	10 fold cross-validation	10 fold cross-validation
	Model parameters	Weka defaults ²¹	Weka defaults
	Estimated Model Accuracy	Use estimated model accuracy	Use estimated model accuracy

Discussion: ❶ In this particular circumstance, no hard and fast rule or heuristic rule exists for determining whether outlier values should be removed or estimated. In fact, human experts claim that such decisions must be carried out on a case by case basis. Hence, for this particular example, though the original case attribute recommended a removal of all outlier values (an additional case attribute mentioned that only few outlier values were present), the better decision would be to estimate the outlier values using the mean operator (since a significant amount of outliers are present and probably represent a default value). Hence, though not ideal for novice data miners, a user would have to make a decision strictly based on the implicit case information (without a rule-based recommendation).

²⁰ For the sake of brevity, DP shall be synonymous for recommendations arising in both the Data Understanding (DU) and Data Preparation (DP) phases, as these are strongly interrelated.

²¹ In order to simplify our results and DM activities, we have opted to use the defaults recommended by the particular DM toolkit (i.e. Weka, Oracle Data Miner)

5.2.2 A Class Imbalance Example

For this example, the data was drawn from a database containing information pertaining to the student admissions process. The dataset consists of approximately 1600 examples, 7 attributes and a nominal target class. The DM objective consists in producing a classifier for predicting student admission status, such as whether a student is admitted on a full-time or part-time basis. Some of the useful dependent attributes for this problem are the program of study, the level of study such as undergraduate or graduate level and the college performance rating. A class imbalance problem is typically characterized by an unusually high model accuracy (i.e. 98% or higher), combined with a class distribution having a disproportionate representation of class values. For example, a 2-class problem where 90% of the labels are in the first category (majority class), and the remaining are labeled with the second category (minority class). Class-imbalance problems are popular in application areas where less frequently occurring events are of interest to the DM problem such as credit card fraud detection. In this particular example, the vast majority of the dataset contains student records which have an admission status as "full-time" (majority class), however the minority class values ("part-time") are considered equally important to our investigation. Figure 40 below illustrates the results from the initial case retrieval phase. In this case, selection of the appropriate similar case, is evidently **case no. 1** since it scores highest for both GSM and GUM values.

Step 3: Selecting a Basis Case for the Current DM Problem						
Case ID	Name	DU	DP	DM	GUM	GSM
1	Predicting Student Program Status	0.2	0.5	0.2	0.9	0.91
4	Predicting Student Status Change Graduation Year	0.07	0.17	0.1	0.334	0.835
3	Predicting Reasons for Student Program Status Change	0.1	0.4	0.2	0.7	0.767

Figure 40 Selecting a Basis Case for a Class Imbalance Example

Table 7 below summarizes the recommendations that were provided by the system during the DM activity:

Table 7 Summary of Recommendations for a Class Imbalance Example

DM Phase	Case Attribute	System Recommendation	Human Expert Recommendation
DP	Feature Reduction Method	Subset Selection	Subset Selection
	Outlier Values Handling	Non-applicable	Non-applicable
	Missing Values Handling	Non-applicable	Non-applicable
	Duplicate Values Handling	Eliminate, but recommendation to carefully evaluate ❶	Eliminate
DM	Model Selection	Use Cost-Sensitive Classifier with cost-matrix and heuristics for creating cost matrix	Use a Cost-Sensitive Classifier and cost-matrix. (use NaiveBaye's as base classifier) ❷
	Train and Test Options	10 fold cross-validation	10 fold cross-validation
	Model parameters	Weka defaults	Weka defaults
	Estimated Model Accuracy	Use F1 measure	Use F1 measure

Discussion: ❶ This situation is similar to the previous example with respect to the handling of outlier values. Though for this particular example, the similar basis case happens to provide the correct answer, the user must proceed with caution. This is an existing limitation of the DM assistant in that it can only provide a cautionary note and not explicitly make a decision on the user's behalf for handling outliers. ❷ It is worth mentioning that though our DM assistant does detect when a class-imbalance may be eminent (i.e. if minority class is less than 5% and estimated model accuracy is unusually high) and provides recommendations about how to implement a cost matrix, this does not ensure a successful mining activity even for an expert data miner. Issues related to imbalanced datasets are complex and still an area of active research. Nonetheless, we believe that providing recommendations is a good start in the right direction. Secondly, though a human expert (through trial and error) was able to assess the best base classifier to resolve this problem, there is no hard and fast rule for

assessing the required base classifier and a novice-data miner must invest the same effort to obtain similar results.

5.2.3 A Feature Reduction Example

In this example, we have mined a data source from a student survey that was compiled in 1996. The dataset consists of approximately 2000 examples, 15 attributes and a nominal target class. The DM objective consists in producing a classifier for predicting the status for graduated students (i.e. employed with children, etc.). Table 8 below summarizes the recommendations that were produced by the system during the DM activity:

Table 8 Summary of Recommendations for a Feature Reduction Example

DM Phase	Case Attribute	System Recommendation	Human Expert Recommendation
DP	Feature Reduction Method	Use PCA (since data is not sparse) ①	Use PCA (since data is not sparse)
	Outlier Values Handling	Remove outliers but recommendation to carefully evaluate ②	Replace with mean
	Missing Values Handling	Non-applicable	Non-applicable
	Duplicate Values Handling	Eliminate but recommendation to carefully evaluate ②	Eliminate
DM	Model Selection	RBFnetwork	J48 ③
	Train and Test Options	10 fold cross-validation	10 fold cross-validation
	Model parameters	Weka defaults	Weka defaults
	Estimated Model Accuracy	Use estimated model accuracy and recommendation to examine more relevant attributes to try to improve accuracy ④	Use estimated model accuracy and recommendation to examine more relevant attributes to try to improve accuracy

Discussion: ❶ Though this recommendation is very useful, a key problem remains for properly assessing whether the problem dataset is sparse or not. See Section 6.1 for additional details on possible future enhancements. ❷ As was previously discussed in both of the above examples, the handling of outlier and duplicate values is a delicate affair. Though the DM assistant provides implicit knowledge from the attribute value of a previously resolved case and a suitable recommendation, the user must still use careful judgement in resolving this issue. ❸ In this particular situation, the recommended model did not yield the best accuracy or performance (67% for RBFnetwork versus 71% for J48). Invariably, this situation may occur due to insufficient case base coverage. ❹ In actual fact, this specific example is a perfect case where the DM activity confirms that a successful model (i.e. a model with an accuracy of at least 75%) cannot be achieved. This most probably resulted because the variables of interest and the target class are naturally uncorrelated.

5.2.4 A Regression Example

For this example, we have mined a data source from an operations database containing student information. The dataset consists of approximately 35000 examples, 5 attributes and a numerical target class. The DM objective consists in producing a regression formula for predicting the student grade point average. Table 9 below summarizes the recommendations that were provided by the system during the activity:

Table 9 Summary of Recommendations for a Regression Example

DM Phase	Case Attribute	System Recommendation	Human Expert Recommendation
DP	Feature Reduction Method	Feature Ranking ❶	Non Required
	Outlier Values Handling	Eliminate but recommendation to carefully evaluate	Eliminate
	Duplicate Handling	Eliminate but recommendation to carefully evaluate ❷	Eliminate
	Examples Reduction	Non-applicable	Non-applicable

DM	Model Selection	NaiveBayes, but recommendation to use a model supporting numerical target ③	SVM
	Train and Test Options	Perform 3 verifications to ensure linear model assumption holds ④	Perform 3 verifications to ensure linear model assumption holds
	Model parameters	Non-applicable	ODM defaults
	Estimated Model Accuracy	Use p-value and correlation coefficient to evaluate. P-value is not less than 5%, poor model ⑤	Use p-value and correlation coefficient to evaluate. P-value is not less than 5%, poor model.

Discussion: ① The *Feature Reduction Method* for the previous basis case recommended the use of a feature ranking method. During the design of the system, we decided not to recommend the use of a feature reduction method unless the quantity of attributes is reasonably large (i.e. 15) so as to avoid the risk of the user accidentally eliminating useful attribute information. Unfortunately, there is no deterministic rule for applying feature reduction. Hence, for this specific decision, it was best to leave it up to the user to interpret the previous case information without assistance. It is worth mentioning however that applying a feature ranking method is harmless as it only provides a recommendation of which attributes are more strongly correlated. The final decision of whether to comply with the ranking recommendation is up to the user. ② In this particular case, there was only a single outlier value for the *etat_pgm* attribute, hence elimination was the correct recommendation. ③ For the *Selected Model* attribute, though the similar basis case used originally was resolved using a *NaiveBayes* model, the system recommended the selection of a new model that can handle a numerical target class. During the elicitation of DM rules, since many models are available that support numerical targets (i.e. ANN, SMO, SVM) and the user could potentially discretize the target class to benefit from the use of other models, we chose to only give a general recommendation (ensure that the chosen model supports

the correct target class data type (i.e. nominal or numerical). ❹ For this decision, recommendations involved explaining how to verify the linear tendency using a dispersion diagram and the use of a histogram to ensure the normality of the obtained residue errors. ❺ The system provided two recommendations for this attribute: a) a p-value should be used since selected DM toolkit only supports a statistical-based approach for regression; b) a warning that the model performance may be inadequate since the p-value is greater than 5% (the result was actually 8%, hence the null hypothesis cannot be rejected).

5.3 A Brief System Evaluation

Although the set of assisted DM examples presented above give some indication as to the assistance capabilities of our system, these by no means exhaust all the possible recommendations (and associated combinations of situations) that our elicited DM knowledge can provide. For instance, though the knowledge within both our DM assistant (i.e. CBR and ontology) is somewhat quantifiable (i.e. 97 concepts, 58 properties, 63 individuals, 68 rules, 40 recommendations, 66 attributes and 3 seed cases), due to combinatorial effects, verifying every possible permutation under which this knowledge can come into play is practically impossible. Hence, faced with such a challenge, it became apparent during the course of our research, that a thorough and quantitative evaluation of our DM assistant would be very difficult. Hence, we have rather opted for an empirical and qualitative evaluation based on a series of DM examples as elaborated in the previous section. As a result, Table 10 provides a concise, yet partially subjective, system evaluation of our DM assistant. Essentially, the table attempts to illustrate if the above DM examples (or tests) provide sufficient evidence that our research objectives have been achieved. The evaluation has been carried out using a simple rating or score such as *Poor*, *Undecided*, *Fair* and *Strong* for each research objective. A brief discussion follows for providing additional insight into our evaluation process.

Table 10 Qualitative System Evaluation

Research Objective	Qualitative Verdict
Support for Non-Experts	Strong
Fostering Knowledge Reuse	Fair
Beyond Model Selection Support	Strong
A Need for Detailed DM Knowledge	Strong

Discussion: The DM assistant scored a *Strong* value for providing support for non-expert data miners in large part because it is fair to assume that the holistic approach (in the form of a CRISP-DM driven case vocabulary) provided by the system, provides better support for novices than the typical use of wizard-like interfaces (as is typically done for most DM toolkits). In addition, aside from the fact that we elicited a rule-set of detailed DM knowledge from sources targeted at novice data miners (i.e. text books), we have also strived to offer recommendations that present definitions of basic terms and concepts before giving a precise directive favors support for non-expert users. We believe this approach favors support for non-expert users. Concerning the fostering of knowledge reuse, we rated the attainment of this objective as *Fair*. This was primarily due to the fact that, though our CBR component provides a basis case for encouraging knowledge reuse, caution must be exercised when re-using case attributes when no recommendations are available. This has been demonstrated numerous times by the above examples when dealing with missing, outlier and duplicate values. As for offering support beyond model selection, we believe that (as demonstrated by some of the examples above) our proposal for extending the meta-learning problem to encompass beyond data characteristics (but rather a holistic DM problem characterization that spans the major phases of the CRISP-DM methodology) provides sufficient evidence that the DM assistant satisfies this research objective. As all of the above DM examples have demonstrated, the use of a complementary knowledge base in the form of rules (detailed DM knowledge) that provide textual recommendations is imperative for supporting non-expert data miners during their DM activities. Last, it is important to put forward that under ideal circumstances (i.e. having had more time and access to a population of independent DM users) it would have proved interesting to perform a

more exhaustive evaluation to assess the level of user satisfaction based on more precise criteria (i.e. usability and interpretability of DM recommendations per user).

CHAPTER 6

Future Directions

This chapter presents some potentially useful and interesting future research directions from which the basis of our current work may be extended. Although we offer many directions, this by no means reflects that the current state of our work was unsatisfactory or incomplete. On the contrary, these only demonstrate that research into data mining assistant technology is a fertile area where many new advancements and discoveries are possible.

6.1 Improving Problem Characterization

Due to time limitations during our research we have only implemented a small subset of data characteristic measures (or indexes) that can be used for aiding the retrieval process of the CBR sub-system. It could prove interesting to incorporate additional data characteristics measures. For example, additional statistical and information-theoretic measures could be implemented for improving the characterization of our DM problem case. Examples of additional statistical measures are the *Correlation Coefficient*, *Variation Coefficient*, and *Covariance* for the attributes and target class of a problem dataset. With respect to information-theoretic measures, DM problem characterization could benefit from additional measures such as the *Normalized Attribute Entropy*, *Class/Attribute Joint Entropy* and *Signal-to-Noise* measures. For further details on such data characteristic measures see Henery [40] and Castiello *et al.* [19].

Though we currently support the automatic computation of 11 out of 14 problem characteristics used for characterizing a DM case, it could prove useful to automate the computation for the presence of *outliers* and *inconsistent* values. Automating the detection of *inconsistent* values is a reasonably trivial problem since it strictly depends on access to supplementary constraint information for each attribute for a given problem set (i.e. permissible data ranges). Although the issue for automating outlier detection is more complex, many approaches from the areas of machine-learning and

statistics have been proposed (i.e. proximity-based, cluster-based and density-based approaches). Due to space constraints, we defer the interested reader to Tan *et al.* [104] for a comprehensive treatment.

Last, it may be interesting to consider automating the computation of other DM case attributes such as the degree of sparseness²² for a given DM problem dataset. For example, we have currently defined a manually selectable boolean case attribute that can prove useful for assessing whether to use the PCA method (over the SVD method) for feature reduction under circumstances where the problem dataset is sparse. For example, the following rule was used for expressing such detailed DM knowledge:

```
DP-03 := DPadvice(pc, featurereduction) ^  
symAttributesRatio(pc, 0.0) ^ sparseData(pc, "no") ^  
featureSelection(pc, ?x2) ^ swrlb:notEqual(?x2,  
"PrincipalComponentAnalysis") -> DPadvice(ac, pca)
```

Essentially, the above rule expresses that if feature reduction advice has been asserted, only numerical attributes are used, the problem dataset is “sparse” (and PCA has not already been chosen), then the final advice is to use the PCA method.

6.2 Beyond Classification Support

As previously mentioned in Section 2.1.4 the majority of research on model selection assistants has exclusively focused on supporting classification DM activities ([3], [4], [5]). We believe that it is unrealistic to constrain data mining assistance exclusively for classification problems. Hence, new research initiatives are required to define effective methods (i.e. data characterization) for intelligently supporting clustering, association and anomaly detection mining DM activities.

²² Sparseness is associated with the degree to which a matrix (or Euclidean space) contains a large number of 0 or undefined values.

6.3 CBR to Ontology Knowledge Promotion

It has been our aim early on in the project (via the use of a CBR-based component) to attempt to use knowledge representation formalisms that can minimize, when possible, the difficulties associated with traditional knowledge solicitation efforts (i.e. ontology and rule-based systems). Our work has established that the case-based reasoning paradigm provides a good basis for the efficient acquisition of data mining knowledge. Nonetheless, the case-based reasoning system does require a complementary knowledge source (a formal DM ontology) in order to fulfill its DM assistance requirements. Not surprisingly, ontologies, like traditional rule-based systems, suffer from the infamous “knowledge acquisition bottleneck”. Hence, new mechanisms are needed to facilitate the ontology knowledge elicitation effort by using complementary knowledge sources (i.e. a CBR system). Hence, it could prove very interesting to investigate new mechanisms for transferring or “promoting” tacit DM knowledge that accumulates within the CBR into a more structured and formalized representation within the DM ontology. One approach might be to data mine (use association mining) the cases in order to discover new patterns or relations that could be subsequently represented or promoted within the ontology as rules. Another approach might involve mining or applying natural language processing techniques to extract new terminology within the free-form text case attributes (DM attributes that allow a user to enter free-form text) in order to derive new data mining ontology concepts. Figure 35 illustrates a modified DM assistant architecture that contains a *Promotion Interface* component. This component could provide the aforementioned data mining and/or natural language processing functionalities and could be used periodically (with the help of a DM *knowledge expert*) during a maintenance cycle for scouring the DM case base and promoting new useful knowledge within our DM ontology.

6.4 Case Maintenance Certification

As previously discussed, due to the complex nature of our DM case representation, the periodic CBR maintenance cycle was delegated to a committee of experts that was responsible for manually assessing the quality and competence of the DM case base (by eliminating potentially harmful cases). It may be interesting to investigate new

knowledge-based methods (possible ontology and/or rule-based) for semi-automatically carrying out case maintenance activities (i.e. detection of harmful cases).

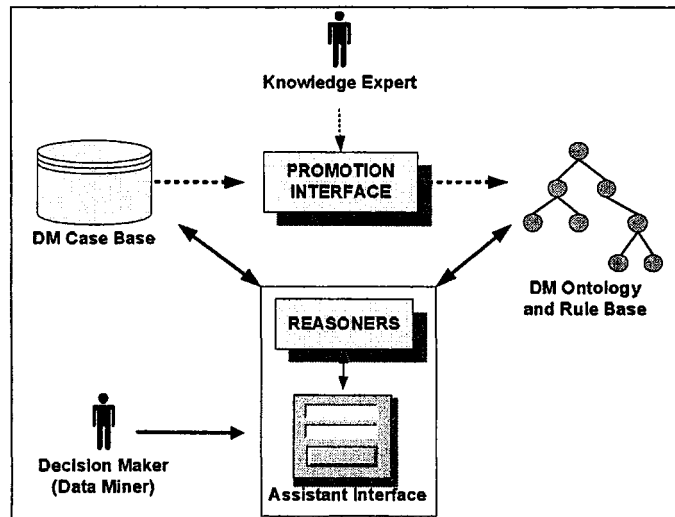


Figure 41 CBR to Ontology Knowledge Promotion

6.5 Leveraging DL Reasoning during Case Adaptation

It might be interesting to investigate how a DL-based reasoner could actively be used during the case adaptation phase in order to target specific levels of detailed DM knowledge depending on the user's level of DM expertise. For instance, since the declarative portion of our elicited detailed DM ontology knowledge is hierarchical (taxonomic) in nature, an expert DM user may be satisfied with getting general advice such as "Apply a Dimensionality Reduction technique" (defined as a parent ontology concept), while a less experienced user may wish a specific recommendation (defined as a more specialized or inherited ontology) for using a particular technique such PCA (Principal Component Analysis).

6.6 Ontology-Based Visual Explorer

It may be useful to make our DM ontology knowledge more explicit to the user. For example, it could prove helpful to make the DM ontology concepts (and intrinsic taxonomy) available to a user during the DM process using a visual or graphical

technique. Evidently, research would need to be carried out in order to assess when it is most appropriate and in what form the DM ontology knowledge could be presented to the user during a DM activity so as to empower the user to make more informed decisions. This mechanism may be more appealing and intuitive for intermediate DM users who find it useful to explore related DM concepts during the user's overall DM learning experience. In this particular case, we are referring to the user's learning experience and not the machine learning experience that may be used to resolve the particular DM problem at hand.

6.7 Improving the Case Adaptation Interface

As previously mentioned during our survey of assisted DM problems (see Section 5.2), though our current case adaptation interface provides a user with recommendations and heuristics in the form of textual messages (located within an adjacent window pane), it is not always obvious for the user which associated case attribute requires modification or attention. Hence, it might be interesting to improve the usability of our case adaptation interface by providing dynamic indicators or arrows which hint to the user where a modification is required. In addition, it may be useful to provide another color indicator for attributes which have been changed and for case attributes which still require evaluation (a possible change) during the on-going DM process.

6.8 Ontological DM Body of Knowledge

As previously discussed in Section 3.5.3, the successful use of ontologies within diverse application fields over the past several years motivates one to consider the tremendous benefits that could result for the data mining community should a similar effort be undertaken. Hence, though much work remains to be done, the current state of our work with ontologies in the area of data mining assistance provides a potential stepping stone for carrying forward subsequent research with the aim of producing a comprehensive and exhaustive ontological DM body of knowledge for many to use.

6.9 Integration with Data Warehousing

Though the potential benefits for data mining that have resulted from the use of data warehousing technology and research can never be undermined, it might be interesting to pursue new research avenues that can leverage the synergistic use of data warehousing and ontologies within the context of intelligent DM assistance. For instance, during the DM process, it often becomes important to provide traceability of how the original data source is progressively transformed. Such operations can result from the application of various data quality and suitability operations such as binarization, normalization and replacement of missing values within a problem dataset. In a large-scale corporate setting, where data warehousing and data mining activities are rarely carried out by the same individual, critical early pre-processing activities which occurred at the data-warehousing end, may not subsequently be made available to a data miner during future DM activities. The loss of such meta-knowledge or insight about a dataset can significantly affect the final outcome of a DM task. A DM user may not be made aware that a problem dataset's missing values were previously replaced with mean/mode values at the data warehousing end, when such values actually represent normal events that could yield valuable insight when interpreted in a proper DM context. Hence, it may be interesting to investigate how an ontology could be devised to hold such "meta-knowledge" and be shared by data warehousing and data mining environments in order to improve the quality of DM efforts.

6.10 Supporting the Deployment Phase

Though our research has focused mainly on the first 5 phases of the CRISP-DM process, it may be interesting to investigate how extending our DM case representation to include knowledge about how DM models are deployed within an active environment. Evidently, this would involve managing DM tasks throughout a complete life cycle. Particularly, it may be interesting to collect and manage additional DM case attributes on the performance, usability and usefulness for a given DM model and see how these could affect the overall CBR retrieval process. In a sense, such added deployment knowledge could re-enforce our existing global utility measure (GUM).

6.11 Overall System Performance Using Utility

Early on during our research, we ascertained that evaluating the overall performance of our DM assistant was a non-trivial task. Though performance metrics such as precision and recall measures are commonly used with information retrieval systems, such measures are not-applicable in our context. As previously mentioned the problem primarily stems from the fact that the solution part of our DM case structure is complex and multi-valued (i.e. the case solution holds data preparation, data modeling and evaluation attributes instead of a trivial single value). Nonetheless, with a sufficient number of DM cases (i.e. greater than 100), it may be interesting to investigate if the global utility measure (associated with each DM case) could be used as an overall system performance metric. For example, since each DM case within the case base has a normalized GUM value between 0 and 1, a “perfect” case base holding 100 DM cases would score a total of 100. Unfortunately, due to time constraints we were not able to acquire a sufficiently high case count in order to consider this approach as a credible means for assessing overall system performance.

CHAPTER 7

Conclusions

Our research journey began with a desire to leverage AI technologies in order to bridge an alarming “chasm” that exists between decision makers and their effective use of data mining technology – a metaphorical view we have coined as the “Decision Support and Data Mining Paradox”. Nowadays decision makers invariably need to use DM technology to tackle complex decision making problems, however the successful application of DM technology requires that one possess specific DM decision making skills. Hence, it has been our goal to put forward a theoretical, conceptual, and technological framework for the realization of an intelligent data mining assistant, capable of empowering non-specialist data miners and potentially help to bridge this DM-DS chasm. Specifically, we have verified if the use of a case-based reasoning system and a formal DL ontology using SWRL rules can provide such an environment for supporting the DM decision making process.

In summary, inspired by the current state-of-the-art in data mining assistance research, we have addressed the following key DM challenges or objectives:

- **Support for the Non-Expert Data Miner** – Current DM research is largely based on the use of very specialized statistics and machine learning techniques.
- **Fostering Knowledge Reuse** - Current DM processes make very little use of existing DM knowledge in the form of “experiences” that can be reused.
- **Beyond Model Selection Support** - Previous research efforts into DM assistants have primarily focused on providing a user with model selection support.

- **A Need for Detailed DM Knowledge** - Existing DM methodologies provide general directives, but non-specialists need explanations and recommendations on how to carry out a DM methodology.

With respect to the above challenges, though a thorough quantitative evaluation of our hybrid intelligent data mining assistant proved very difficult, the results section of this report (Section 5) has provided convincing evidence in the form of assisted DM examples and a qualitative evaluation that the aforementioned DM objectives have been satisfied to a reasonable degree.

Perhaps our research efforts can best be summarized by presenting the novel features or key benefits that have provided our intelligent DM assistant:

- Having demonstrated the power of representing and capturing a DM activity as a case or “experience” using the CBR paradigm.
- Extending the traditional meta-learning problem to encompass DM problem characteristics (i.e. missing, duplicate, inconsistent and outlier values) has provided an efficient mechanism for retrieving and reusing a DM experience from a case base.
- The combined use of similarity-oriented (GSM) and a utility-oriented (GUM) measures during the retrieval phase has provided a means for improving initial case retrieval for novice data miners of a CBR system.
- The use of two complementary knowledge bases (i.e. CBR and formal ontology) has proven very effective for supporting the DM case adaptation process and invariably helping the novice user formulate a solution to a given DM problem.
- The use of detailed DM knowledge in the form of a rule-set has not only proven very effective for supporting novice data miners by offering recommendations and heuristics, but has also provided a means for bridging two initially disparate knowledge source (i.e. the CBR and the formal ontology).

In addition, as previously mentioned in Section 2.2, since the effective representation of procedural knowledge within ontologies is currently an active and much debated area of research, we hope that our having successfully elicited and applied detailed DM knowledge in the form of procedural rules (along with the aforementioned future directions stated in Chapter 6) shall encourage further new research initiatives within this area.

On a more philosophical note, though data mining is not founded on a solid theoretical framework from first principles (and from the fact that the daily practice of DM is a constant reminder that the field is unyieldingly more of an "art" than a formal science), we firmly believe that the realization of our intelligent DM assistant has established that the use of knowledge-based systems within the area of DM assistance holds great promise and potential.

Appendix A - Description Logic Classification

Table 11 Description Logic Classification Symbols

DL symbol	Concept or Role Operator Support
S	supports standard connectives (\wedge, \vee, \neg) and quantifiers (\forall, \exists) . equivalent to <i>ALC</i> with transitive roles (<i>R+</i>)
H	supports role inclusion axioms (i.e. role hierarchy)
O	supports nominals (i.e. singleton classes)
I	supports inverse roles
N	supports number restrictions
Q	supports qualified number restrictions
D	Support data types

Appendix B - DM Problem Characteristics Details

The following contains additional details in the form of permissible ranges and mathematical formulas for the DM problem characteristics that were used for the implementation of our DM assistant.

Table 12 Problem Characteristics (Feature Indexes)

#	Problem Characteristic	Detailed Description
1	Business Area	(Admission, retention, follow-up)
2	DM Activity Type	(Classification, Regression)
3	Number of Examples	Integer (1 ... 8000)
4	Number of Attributes	Integer (1 ... 25)
5	Number of Classes	Integer (1 ... 8000)
6	Mean Skewness	$mean[\forall_K \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{\sigma_X^3} \frac{\sum_{k=1}^K (x_k - \bar{X}_m)^3}{K} \right)]$
7	Mean Kurtosis	$mean[\forall_K \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{\sigma_X^4} \frac{\sum_{k=1}^K (x_k - \bar{X}_m)^4}{K} \right)]$
8	Normalized Class Entropy	$\overline{H(C)} = \frac{H(C)}{\log_2(n)} = \frac{\sum_{i=1}^n \pi_i \log_2(\pi_i)}{\log_2(n)}$
9	Maximum Mutual Information	$MI(C, X) = \max[\forall_x \left\{ \frac{1}{m} \sum_{i=1}^m \left(\sum_{j=1}^K \sum_{l=1}^m p_{ij} \log_2 \left(\frac{p_{ij}}{\pi_i q_j} \right) \right) \right\}]$
10	Target Data Type	(Numerical, Nominal)
11	Ratio of Duplicate Examples	$\frac{\sum_{i=1}^r \sum_{j=1}^m x_{i,j} = NULL}{Km}$

12	Has Outlier Values	(yes, no)
13	Ratio of Missing Values	$\frac{\sum_{i=1}^r \sum_{j=1}^m x_{i,j} = NULL}{Km}$
14	Has Inconsistent Values	(yes, no)

Appendix C - List of Data Mining Case Attributes

The following is a complete list of the 66 features that were used to represent a DM case. Indexes are indicated by a star symbol (*).

Table 13 Data Mining Case Attributes

PHASE	ATTRIBUTE
Business Understanding	Case name Business problem Business area * Business success criteria Potential data sources Data mining problem Expected model accuracy Data mining activity type Tools assessment
Data Understanding	Data Location Available data format Attribute List Target Class Attribute DataTypes Transactional Data Number of examples * Number of attributes * Number of classes * Ratio of symbolic attributes * Mean skewness * Mean kurtosis * Norm class entropy * Max mutual information * Target data type * Minority Class Percentage Sparse Data Duplicate Example Ratio * Has outlier values * Missing Values Ratio * Has inconsistent values *
Data Preparation	Feature Reduction Method Feature Reduction Details Example Reduction Method Duplicates Handling Duplicates Details Outliers Handling Outliers Details Missing Handling

	Missing Details Inconsistent Handling Inconsistent Details Feature Creation Method Feature Creation Details Data Aggregations Performed Aggregation Details Data Transformation Transformation Details
Data Modeling	Modeling assumptions Selected model Train and test options Model Parameters Other Model Parameters Estimated Model accuracy Confidence interval Limits F1-measure or AUC Root Mean Squared Residue Error Correlation Coefficient (r) Probability of Error (p-value) DM Success Criteria Achieved Final Model Location
Process Evaluation	Business success criteria achieved DU local utility score DP local utility score DM local utility score Improvement suggestions

Appendix D - List of Elicited SWRL Rules

The following presents an unabridged list of the 68 SWRL rules which were crafted for providing the detailed DM knowledge of our system. The rules are expressed in the form of antecedent-consequent pairs. For the sake of brevity, the individuals containing annotated text properties (actual recommendation text) are not shown.

1) BUSINESS UNDERSTANDING RULES:

```
BU-01 := expModelAccuracy(pc, ?x) ^ swrlb:greaterThan(?x, 90.0) -> BUadvice(ac,
overfitting)
BU-02 := toolAssessment(pc, "Weka") ^ DMActivityType(pc, "regression") ->
BUadvice(ac, mlapproach)
BU-03 := toolAssessment(pc, ?x2) ^ swrlb:notEqual(?x2, "Weka") ^
DMActivityType(pc, "regression") -> BUadvice(ac, statapproach)
BU-04 := expModelAccuracy(pc, ?x1) ^ swrlb:greaterThan(?x1, 1.0) ^ advice(ac,
statapproach) -> BUadvice(ac, pvalue)
```

3) DATA UNDERSTANDING & DATA PREPARATION RULES:

feature reduction:

```
DP-01 := noAttributes(pc, ?x1) ^ swrlb:greaterThan(?x1, 2) ^
swrlb:lessThan(?x1, 20) ^ featureSelection(pc, "non-applicable") ->
DPadvice(ac, featureselection)
DP-02 := noAttributes(pc, ?x1) ^ swrlb:greaterThan(?x1, 19) ^
swrlb:lessThan(?x1, 60) -> DPadvice(pc, featurereduction)
DP-03 := DPadvice(pc, featurereduction) ^ symbAttributesRatio(pc, 0.0) ^
sparseData(pc, "no") ^ featureSelection(pc, ?x2) ^ swrlb:notEqual(?x2,
"PrincipalComponentAnalysis") -> DPadvice(ac, pca)
DP-04 := DPadvice(pc, featurereduction) ^ symbAttributesRatio(pc, 0.0) ^
sparseData(pc, "yes") ^ featureSelection(pc, ?x2) ^ swrlb:notEqual(?x2,
"SingularValueDecomposition") -> DPadvice(ac, svd)
DP-044 := DPadvice(pc, featurereduction) ^ symbAttributesRatio(pc, ?x1) ^
swrlb:greaterThan(?x1, 0.0) -> DPadvice(ac, discretizenominals)
DP-05 := minorityClassPerc(pc, ?x1) ^ swrlb:lessThan(?x1, 0.05) ^
exampleSelection(pc, "random sampling") -> DPadvice(ac, stratifiedsampling)
outlier and missing robustness:
DP-07 := selectedModel(pc, "naiveBayes") -> isOutlierRobust(ac, "yes")
DP-08 := selectedModel(pc, "id3") -> isOutlierRobust(ac, "no")
```

```

DP-09 := selectedModel(pc, "j48")           -> isOutlierRobust(ac, "no")
DP-10 := selectedModel(pc, "kNearestNeighbor") -> isOutlierRobust(ac, "no")
DP-11 := selectedModel(pc, "linearRegression") -> isOutlierRobust(ac, "no")
DP-12 := selectedModel(pc, "logisticRegression") -> isOutlierRobust(ac, "no")
DP-13 := selectedModel(pc, "RBFnetwork")     -> isOutlierRobust(ac, "no")
DP-14 := selectedModel(pc, "SMO")           -> isOutlierRobust(ac, "no")
DP-15 := selectedModel(pc, "naiveBayes")     -> isMissingRobust(ac, "yes")
DP-16 := selectedModel(pc, "id3")           -> isMissingRobust(ac, "no")
DP-17 := selectedModel(pc, "j48")           -> isMissingRobust(ac, "no")
DP-18 := selectedModel(pc, "kNearestNeighbor") -> isMissingRobust(ac, "no")
DP-19 := selectedModel(pc, "linearRegression") -> isMissingRobust(ac, "no")
DP-20 := selectedModel(pc, "logisticRegression") -> isMissingRobust(ac, "no")
DP-21 := selectedModel(pc, "RBFnetwork")     -> isMissingRobust(ac, "yes")
DP-22 := selectedModel(pc, "SMO")           -> isMissingRobust(ac, "no")

data quality issues handling:
DP-23 := hasOutliers(pc, "yes") ^ isOutlierRobust(ac, "no") ^
outlierHandling(pc, "non-applicable") -> DPadvice(ac, outlier)
DP-24 := missingValuesRatio(pc, ?x1) ^ swrlb:notEqual(?x1, 0.0)
isMissingRobust(ac, "no") ^ missingHandling(pc, "non-applicable") ->
DPadvice(ac, missing)
DP-25 := hasInconsistents(pc, "yes") ^ inconsistentHandling(pc, "non-
applicable") -> DPadvice(ac, inconsistent)
DP-26 := duplicateExamplesRatio(pc, ?x1) ^ swrlb:notEqual(?x1, 0.0) ^
duplicateHandling(pc, "non-applicable") -> DPadvice(ac, duplicates)

elimination warnings:
DP-30 := duplicateHandling(pc, "eliminate") -> DPadvice(ac,
eliminatewarning)
DP-31 := outlierHandling(pc, "eliminate") -> DPadvice(ac,
eliminatewarning)
DP-32 := missingHandling(pc, "eliminate") -> DPadvice(ac,
eliminatewarning)
DP-33 := inconsistentHandling(pc, "eliminate") -> DPadvice(ac,
eliminatewarning)

model constraints:
DP-34 := selectedModel(pc, "id3") -> DPadvice(pc, requirenomtarget)
DP-35 := selectedModel(pc, "j48") -> DPadvice(pc, requirenomtarget)
DP-36 := selectedModel(pc, "linearRegression") -> DPadvice(pc,
requirenumtarget)
DP-37 := selectedModel(pc, "logisticRegression") -> DPadvice(pc,
requirenomtarget)

```

```

DP-38 := selectedModel(pc, "SMO") -> DPadvice(pc, requirenomtarget)
DP-39 := selectedModel(pc, "naiveBayes") -> DPadvice(pc, requirenomtarget)
binarize, discretize, normalize, aggregation:
DP-40 := transformPerformed(pc, "binarization") ^ DPadvice(pc,
requirenumtarget) -> DPadvice(ac, binarization)
DP-41 := transformPerformed(pc, "discretization") ^ DPadvice(pc,
requirenomtarget) -> DPadvice(ac, discretization)
DP-42 := noExamples(pc, ?x1) ^ swrlb:greaterThan(?x1, 30000) ^
transactionData(pc, "yes") ^ dataAggregationPerformed(pc, "no") ->
DPadvice(ac, aggregation)
DP-43 := transformPerformed(pc, "normalization") -> DPadvice(ac, normalization)
DP-44 := transformPerformed(pc, "applySimpleFunction") -> DPadvice(ac,
appliesimplefunction)
model constraint verification:
DP-45 := transformPerformed(pc, "binarization") -> DPadvice(pc,
nomTonumApplied)
DP-46 := transformPerformed(pc, "discretization") -> DPadvice(pc,
numTonumApplied)
DP-47 := DPadvice(pc, requirenumtarget) ^ targetDataType(pc, ?x1) ^
swrlb:notEqual(?x1, "numerical") -> DPadvice(ac, numericalTargetOnly)
DP-48 := DPadvice(pc, requirenomtarget) ^ targetDataType(pc, ?x1) ^
swrlb:notEqual(?x1, "nominal") -> DPadvice(ac, nominalTargetOnly)
DP-49 := DPadvice(pc, requirenumtarget) ^ targetDataType(pc, "nominal") ^
transformPerformed(pc, "discretization") -> DPadvice(ac, binarization)
DP-50 := DPadvice(pc, requirenomtarget) ^ targetDataType(pc, "numerical") ^
transformPerformed(pc, "binarization") -> DPadvice(ac, discretization)

```

4) DATA MODELING RULES:

```

class imbalance:
GDM-00 := noClasses(pc, 2) ^ minorityClassPerc(pc, ?x1) ^ swrlb:lessThan(?x1,
0.10) -> DMadvice(pc, classimbalance)
GDM-01 := DMadvice(pc, classimbalance) ^ selectedModel(pc, ?x2) ^
swrlb:notEqual(?x2, "costSensitiveClassifier") -> DMadvice(ac,
costsensitivelearning)
model selection train/test options:
GDM-02 := noExamples(pc, ?x1) ^ swrlb:lessThan(?x1, 250) ^ trainTestOptions(pc,
?x2) ^ swrlb:notEqual(?x2, "Bootstrap") -> DMadvice(ac, bootstrap)

```

```

GDM-03 := noExamples(pc, ?x1) ^ swrlb:greaterThan(?x1, 250) ^
trainTestOptions(pc, ?x2) ^ swrlb:notEqual(?x2, "10 Fold Cross Validation") ->
DMadvice(ac, crossvalidation)
assumptions:
GDM-04 := DPadvice(pc, requirenomtarget) ^ modelAssumptions(pc, ?x1) ^
swrlb:notEqual(?x1, "NominalTargetClass") -> DMadvice(ac, nomtargetassumption)
GDM-xx := DPadvice(pc, requirenumtarget) ^ modelAssumptions(pc, ?x1) ^
swrlb:notEqual(?x1, "NumericalTargetClass") -> DMadvice(ac,
numtargetassumption)
model assessment:
GDM-06 := BUadvice(ac, statapproach) ^ pValue(pc, ?x1) ^ swrlb:lessThan(?x1,
0.0) ^ swrlb:greaterThan(?x1, 0.05) -> DMadvice(ac, pvalue)
GDM-07 := BUadvice(ac, mlapproach) ^ residueErrors(pc, ?x1) ^
swrlb:lessThan(?x1, 0.0) -> DMadvice(ac, residue)
model assessment
GDM-08 := DPadvice(pc, requirenomtarget) ^ actualModelAccuracy(pc, ?x1) ^
swrlb:lessThan(?x1, 75.0) ^ dmSuccessCriteriaAchieved(pc, ?x2) ^
swrlb:notEqual(?x2, "poor") -> DMadvice(ac, poorDMSuccess)
GDM-09 := DPadvice(pc, requirenomtarget) ^ actualModelAccuracy(pc, ?x1) ^
swrlb:greaterThan(?x1, 74.0) ^ swrlb:lessThan(?x1, 85.0) ^
dmSuccessCriteriaAchieved(pc, ?x2) ^ swrlb:notEqual(?x2, "fair") ->
DMadvice(ac, fairDMSuccess)
GDM-10 := DPadvice(pc, requirenomtarget) ^ actualModelAccuracy(pc, ?x1) ^
swrlb:greaterThan(?x1, 84.0) ^ dmSuccessCriteriaAchieved(pc, ?x2) ^
swrlb:notEqual(?x2, "strong") -> DMadvice(ac, strongDMSuccess)
GDM-11 := DMadvice(pc, classimbalance) ^ fMeasure(pc, ?x1) ^
swrlb:lessThan(?x1, 0.0) -> DMadvice(ac, flmeasure)
GDM-12 := DPadvice(pc, requirenumtarget) ^ DMadvice(ac, pvalue) ^ pValue(pc,
?x1) ^ swrlb:greaterThan(?x1, 0.05) -> DMadvice(ac, nullhypoexists)
GDM-13 := DPadvice(pc, requirenomtarget) ^ selectedModel(pc, ?x2) ^
swrlb:notEqual(?x2, "costSensitiveClassifier") -> DMadvice(ac, errorrate)
GDM-14 := DPadvice(pc, requirenumtarget) ^ selectedModel(pc, ?x2) ^
swrlb:notEqual(?x2, "costSensitiveClassifier") -> DMadvice(ac, noerrorrate)
GDM-15 := selectedModel(pc, "linearRegression") ^ modelAssumptions(pc, ?x1) ^
swrlb:notEqual(?x1, "NormalLinearModelAssumption") -> DMadvice(ac,
assumelinearmodel)

```

5) PROCESS EVALUATION RULES:

HEV-01 := duScore(pc, "undecided") -> EVadvice(ac, score)

HEV-02 := dpScore(pc, "undecided") -> EVadvice(ac, score)

HEV-03 := dmScore(pc, "undecided") -> EVadvice(ac, score)

References

- [1] A. Aamodt. Explanation-Driven Case-Based Reasoning. in Topics in Case-Based Reasoning. 1994. Berlin: Springer.
- [2] A. Aamodt. Knowledge-Intensive Case-Based Reasoning in CREEK. in Advances in Case Based Reasoning, 7th European Conference. 2004. Madrid, Spain: Springer.
- [3] D. Aha, CBR Resource Page, <http://home.earthlink.net/.../case-based-reasoning.html>.
- [4] L. An, J. Yan, and L. Tong. An Integrated Rule-Based and Case-Based Reasoning System for Customer Service Management. in Proceedings IEEE Conference on e-Business Engineering. 2005.
- [5] K. Bartlmae. Optimizing Data-Mining Processes: A CBR Based Experience Factory for Data Mining. in ICSC. 1999: Springer.
- [6] M. Bauer and S. Baldes. An Ontology-Based Interface for Machine Learning. in Intelligent User Interfaces. 2005. San Diego, California.
- [7] J. Bello-Thomas, P. Gonzalez-Calero, and B. Diaz-Agundo. JColibri: An Object-Oriented Framework for Building CBR Systems. in Advances in CBR, 7th European Conference. 2004. Madrid, Spain: Springer.
- [8] H. Bensusan and C. Giraud-Carrier. Discovering Task Neighborhoods Through Landmark Learning Performances. in 4th European Conference on Principles and Practices of Knowledge Discovery in Databases. 2000.
- [9] R. Bergmann, M. Richter, S. Schmitt, A. Stahl, and I. Vollrath. Utility-Oriented Matching: A New Research Direction for Case-Based Reasoning. in GWCBR-2001 - Proceedings of the 9th German Workshop on Case-Based Reasoning. 2001. Baden-Baden, Germany.
- [10] A. Bernstein, F. Provost, and S. Hill. Intelligent Assistance for the Data Mining Process: An Ontology-based Approach. in IEEE Transactions on Knowledge and Data Engineering. 2005.
- [11] H. Berrer, I. Paterson, and J. Keller. Evaluation of Machine-Learning Algorithm Ranking Advisors. in Data Mining, Decision Support , Meta-Learning and ILP. 2000.
- [12] I. Bichindaritz, Mémoire: Case Based Reasoning Meets the Semantic Web in Biology and Medecine. Advances in Case Based Reasoning, 7th European Conference, LNAI, Spain, 2004.
- [13] K. Borner. Structural Similarity as Guidance in Case-Based Design. in First European Workshop on on Case-Based Reasoning. 1993. Berlin: Springer.

- [14] P.B. Brazdil and J.P. Soares, Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. *Machine Learning*, 2003(50): p. 251-277.
- [15] H. Bunke and B. Messmer. Structural Similarity as Guidance in Case-Based Design. in *First European Workshop in Case-Based Reasoning*. 1993. Berlin: Springer.
- [16] M. Cannataro and C. Camito. A Data Mining Ontology for Grid Programming. in *1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing 2003*.
- [17] D. Caruana, Multitask Learning: Special Issue on Inductive Transfer. *Machine Learning*, 1997(28).
- [18] Caspian, Case-Based Reasoner, <http://www.rverson.ca/~dgrimsha/courses/cps820/CBRCaspian.html>.
- [19] C. Castliello, G. Castellano, and A. Fanelli, Meta-Data: Characterization of Input Features for Meta-Learning. *Modeling Decisions for Artificial Intelligence*, LNAI, 2005: p. 457-468.
- [20] N. Cercone, A. An, and C. Chan. Rule-Induction and Case-Based Reasoning: Hybrid Architectures Appear Advantageous. in *IEEE Transactions on Knowledge and Data Engineering*. 1999.
- [21] M. Charest and S. Delisle. Ontology-Guided Intelligent Data Mining Assistance: Combining Declarative and Procedural Knowledge. in *10th Int. Conference on Artificial Intelligence and Soft Computing*. 2006. Mallorca, Spain.
- [22] M. Charest, S. Delisle, and O. Cervantes. Design Considerations for a CBR-based Intelligent Data Assistant. in *Artificial Intelligence and Software Engineering*. 2006. Agadir, Morocco.
- [23] M. Charest, S. Delisle, O. Cervantes, and Y. Shen. Intelligent Data Mining Assistance via CBR and Ontologies. in *DEXA-Philosophies and Methodologies in Knowledge Discovery*. 2006. Poland: IEEE Computer Society Press.
- [24] CLIPS, A Tool for Building Expert Systems, <http://www.ghq.net/clips/CLIPS.html>.
- [25] CRISP-DM1.0, A Step-by-step Data Mining Guide. 2000, www.crisp-dm.org.
- [26] Cyc, Cyc Upper Ontology, <http://www.cyc.com/cyc/>.
- [27] S. Delisle. Integrating Data Mining and Decision Support via Computational Intelligence. in *DEXA-Philosophies and Methodologies in Knowledge Discovery*. 2005. Denmark: IEEE Computer Society Press.
- [28] B. Diaz-Agudo and P. Conzalez-Calero. An Architecture for Knowledge Intensive CBR Systems. in *European Conference on Case-Based Reasoning*. 2000. Berlin: Springer-Verlag.

- [29] C. Elkan, Magical Thinking in Data Mining: Lessons from Coil Challenge 2000. ACM, 2001.
- [30] EngMath, Ontology for Engineering Mathematics, <http://www-ksl.stanford.edu/knowledge-sharing/papers/engmath.html>.
- [31] J. Federhofer, Medical Expert Systems, <http://www.computer.privateweb.at/judith/index.html>.
- [32] Fionn, Fionn CBR Framework, https://www.cs.tcd.ie/research_groups/mlg/CBML/.
- [33] P. Gama, J. Brazdil, and C. Giraud-Carrier. A Characterization of Classification Algorithms. in 7th Portuguese Conference on Artificial Intelligence. 1995.
- [34] G. Garza and M. Maher. A Process Model for Evolutionary Design Case Adaptation. in Artificial Intelligence in Design 2000. Dordrecht: Kluwer.
- [35] C. Giraud-Carrier, Reporting on the Data Mining Advisor. Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005.
- [36] P. Giudici, Applied Data Mining: Statistical Methods for Business and Industry. 2003: Wiley & Sons.
- [37] C. Golbreich. Combining Rule and Ontology Reasoners for the Semantic Web. in Rules and Rule Markup Languages for the Semantic Web 2004: Springer.
- [38] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, ed. Springer. 2004.
- [39] K. Hammond. CHEF: A Model of Case-Based Planning. in 5th National Conference on Artificial Intelligence. 1990. California: AAAI.
- [40] Henery, Methods of Comparison. Machine Learning, Neural and Statistical Classification, 1994. ch. 7: p. 107-124.
- [41] Hoolet, OWL-DL reasoner <http://owl.man.ac.uk/hoolet/>.
- [42] I. Horrocks, P. Patel-Schneider, and F.v. Harmelen, From SHIQ and RDF to OWL: The Making of a Web Ontology Language. xxx, 2005.
- [43] I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. in Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. 2001. Washington, USA.
- [44] IUCBRF, The IUCBRF CBR Framework, <http://www.cs.indiana.edu/~sbogaert/CBR/>.

- [45] N. Japkowicz. The Class Imbalance Problem: Significance and Strategies. in International Conference on Artificial Intelligence. 2000. Ottawa.
- [46] Java, Java Programming, www.java.com.
- [47] Javascript, Mozilla Scripting Technology, <http://www.javascript.com/>.
- [48] JColibri, Case-Based Reasoning Framework, <http://gaia.fdi.ucm.es/grupo/projects/jcolibri/>.
- [49] Jena, A Semantic Web Framework for Java, <http://jena.sourceforge.net/ontology/>.
- [50] JESS, Java Expert System Shell, <http://herzberg.ca.sandia.gov/jess/>.
- [51] Jython, Jython Programming Language, www.jython.org.
- [52] A. Kalousis and M. Theoharis, NOEMON: Design, Implementation and Performance Results of an Intelligent Assistant for Classification Selection. Intelligent Data Analysis, Elsevier, 1999.
- [53] M. Kantardzic, Data Mining: Concepts, Models, Methods and Algorithms. 2003: IEEE Press, Wiley & Sons.
- [54] KOAN2, Semantic Web Framework, <http://kaon2.semanticweb.org/>.
- [55] R. Kohavi, L. Mason, R. Parekh, and Z. Zheng, Lessons and Challenges from Mining Retail e-Commerce Data. Machine Learning Journal, 2004(Special Issue on Data Mining Lessons Learned).
- [56] D. Leake and D. Wilson. Remembering Why to Remember: Performance-Guided Case Base Maintenance. in Proceedings of EWCBR-2K. 2000.
- [57] B. Leite and P. Brazdil. Improving Progressive Sampling via Meta-Learning on Learning in ECML. 2004: Springer.
- [58] T. Liao, Z. Zhang, and C. Mount, Similarity Measures for Retrieval in Case-Based Reasoning Systems. Applied Artificial Intelligence, 1998. **12**: p. 267-288.
- [59] G. Linder and R. Studer, AST: Support for Algorithm Selection with a CBR Approach. Recent Advances in Meta-Learning and Future Work, 1999: p. 38-47.
- [60] R. Lopez-de-Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, and S. Craw, Retrieval, Reuse, Revision and Retention in Case-Based Reasoning. The Knowledge Engineering Review, 2005.
- [61] LPA, Logic Programming Associates CBR Extension, <http://www.lpa.co.uk/>.

- [62] G. Marakas, Decision Support Systems in the 21st Century. 2 ed. 2003: Prentice-Hall.
- [63] S. Marling, X. Petot, and X. Sterling. Integrating Case-Based and Rule-Based Reasoning to Multiple Design Constraints. in xxx. 1999.
- [64] D. McSherry, Intelligent Case-Authoring Support in CaseMaker-2. Computational Intelligence, 2001. 17(2).
- [65] Metal, A Meta-Learning Assistant for Providing Data Mining Support, www.metal-kdd.org.
- [66] S. Montani and R. Bellazzi. Integrating Case Based and Rule Based Reasoning in a DSS: Evaluation with Simulated Patients. in Proceedings of AMIA Annual Symposium. 1999.
- [67] Mysql, Mysql 4.1 server www.mysql.com.
- [68] R. Neches, R. Fikes, T. Gruber, T. Senator, and W. Swartout, Enabling Technology for Knowledge Sharing. AI Magazine, 1991
- [69] N. Noy and D. McGuinness, Ontology Development 101: A Guide to Creating Your First Ontology, http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html.
- [70] M. O'Connor, H. Knublauch, S. Tu, B. Grosz, M. Dean, W. Grosso, and M. Musen. Supporting Rule System Interoperability on the Semantic Web with SWRL. in ISWC, LNCS 3729. 2005. Berlin.
- [71] OCML, Operational Conceptual Modeling Language, <http://kmi.open.ac.uk/projects/ocml/>.
- [72] OIL, Ontology Inference Language, <http://www.ontoknowledge.org/oil/>.
- [73] Oracle, Oracle 10g Release 2 www.oracle.com.
- [74] Oracle, Oracle Data Miner, <http://www.oracle.com/>.
- [75] OWL, Web Ontology Language, <http://www.w3.org/TR/owl-ref/>.
- [76] S. Pal and S. Shiu, Foundations of Soft Case-Based Reasoning. Series on Intelligent Systems. 2004: Wiley & Sons.
- [77] Pellet, DL Reasoner, <http://www.mindswap.org/2003/pellet/>.

- [78] Y. Peng, P. Flach, P. Brazdil, and P. Soares. Tree-Based Characterization for Meta-Learning. in ECML/PKDD. 2002: Workshop on Integration and Collaboration Aspects of Data Mining.
- [79] J. Phillips and B. Buchanan. Ontology-Guided Knowledge Discovery in Databases. in International Conference on Knowledge Capture. 2001. Victoria, Canada.
- [80] F. Prentzas and E. Hatzilygeroudis. Integrating Hybrid Rule-Based with Case-Based Reasoning. in European Conference on Case Based Reasoning. 2002.
- [81] Protégé-OWL, Protégé OWL editor, <http://protege.stanford.edu/overview/protege-owl.html>.
- [82] Protégé, Protégé OWL API, <http://protege.stanford.edu/plugins/owl/api/guide.html>.
- [83] Racer, DL Reasoner, www.racer.com.
- [84] RDF(S), Resource Description Framework Schema, <http://www.w3.org/RDF/>.
- [85] RDQL, A Query Language for RDF, <http://www.w3.org/Submission/RDQL/>.
- [86] M. Richter. The Knowledge Contained in Similarity Measures. in International Conference on Case-Based Reasoning. 1995.
- [87] ROWL, Rule Extension of OWL Mobile Commerce Lab <http://projects.semwebcentral.org/projects/rowl/>.
- [88] Russel and Norvig, Artificial Intelligence: A Modern Approach. 2 ed. 2003: Prentice-Hall.
- [89] SAS, Enterprise Miner, <http://www.sas.com/>.
- [90] SAS, SEMMA Methodology, <http://www.sas.com/technologies/analytics/datamining/miner/semma.html>.
- [91] J. Schafer, Analysis of Incomplete Multivariate Data. 1997: Chapman and Hall.
- [92] R. Schank, Dynamic Memory: A Theory of Remining and Learning in Computers and People. 1982, New York: Cambridge University Press.
- [93] G. Schreiber, H. Akkermans, A. Anjewierden, R. deHoog, N. Shadbolt, W.V.d. Velde, and B. Wielinga, Knowledge Engineering and Management. 2000: MIT Press.
- [94] Servlet, Java Servlet Technology, <http://java.sun.com/products/servlet/>.
- [95] B. Smyth and P. Cunningham. The Utility Problem Analyzed: A Case-Based Reasoning Perspective. in Avances in Case-Based Reasoning. 1996: Springer.

- [96] B. Smyth and E. McKenna, Competence Models and the Maintenance Problem. Computational Intelligence, 2001. 17(2).
- [97] SPARQL, Simple Protocol and RDL Query Language, <http://www.w3.org/TR/rdf-sparql-query/>.
- [98] R. Studer, V. Benjamin, and D. Fenzel, Knowledge Engineering: Principles and Methods. IEEE Transactions on Data and Knowledge Engineering, 1998.
- [99] SUMO, Suggested Upper Merged Ontology.
- [100] A. Suyama, N. Negishi, and T. Yamaguchi, CAMLET: A Platform for Automatic Composition of Inductive Applications Using Ontologies. Progress in Discovery Science, 2002.
- [101] SweetRules, Tools for Semantic Web Rules and Ontologies, <http://sweetrules.projects.semwebcentral.org/>.
- [102] SWRL-Jess-Bridge, A SWRL Rule Bridge for Using JESS, <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab>.
- [103] SWRL, Semantic Web Rule Language, <http://www.w3.org/Submission/SWRL/>.
- [104] P. Tan, M. Steinbach, and V. Kumar, Introduction to Data Mining. 2005, Boston, MA: Addison-Wesley.
- [105] S. Thrun, Lifelong Learning Algorithms In Learning to Learn. Kluwer Academic Publishers, 1998: p. 181-209.
- [106] Tomcat, Tomcat Web Container, tomcat.apache.org.
- [107] E. Turban, J. Aronson, and T.-P. Liang, Decision Support Systems and Intelligent Systems. 7 ed. 2005, Boston, MA: Pearson.
- [108] UMLS, Unified Medical Language System, <http://umlsinfo.nlm.nih.gov/>.
- [109] Unica, Affinium Model, <http://www.unica.com/product/product.cfm?pw=model>.
- [110] Vampire, Vampire Automatic Theorem Prover, <http://www.knowledgehorizons.manchester.ac.uk/people/index.asp?personID=174>.
- [111] R. Vivalta, C. Giraud-Carrier, P. Brazdil, and C. Soares, Using Meta-Learning to Support Data Mining. Journal of Computer Science Applications, 2004: p. 31-45.
- [112] I. Watson, AI-CBR Resource Page, www.ai-cbr.org.
- [113] I. Watson, Applying Case-Based Reasoning: Techniques for Enterprise Systems. 1997: Morgan Kaufmann.

[114] I. Watson and S. Perera, The Evaluation of Hierarchical Case Representation Using Context Guided Retrieval. CBR Research and Development, LNAI, Springer, 1997.

[115] Weka, Waikato Environment for Knowledge Analysis,
<http://www.cs.waikato.ac.nz/ml/weka/>.

[116] S. Wess, K. Althoff, and G. Derwand. Using K-D Trees to Improve the Retrieval Step in Case-Based Reasoning. in First European Workshop on Case-Based Reasoning. 1993. Berlin.

[117] D. Wettschereck and D. Aha. Weighting Features. in International Conference on Case-Based Reasoning. 1995.

[118] I. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. 2 ed. 2005, San Francisco, CA: Morgan Kaufman.

[119] WordNet, A Lexical Database for the English Language,
<http://wordnet.princeton.edu/>.