A MODULAR ONTOLOGY OF DATA MINING

Panče Panov

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia, July 2012

Evaluation Board:

Prof. Dr. Nada Lavrač, Chair, Jožef Stefan Institute, Ljubljana, Slovenia

Dr. Larisa Soldatova, Member, Brunel University, London, United Kingdom

Prof. Dr. Dunja Mladenić, Member, Jožef Stefan Institute, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Panče Panov

A MODULAR ONTOLOGY OF DATA MINING

Doctoral Dissertation

MODULARNA ONTOLOGIJA PODATKOVNEGA RUDARJENJA

Doktorska disertacija

Supervisor: Prof. Dr. Sašo Džeroski

Ljubljana, Slovenia, July 2012

Contents

To the memory of my grandfather Bruno

Abstract

The domain of data mining (DM) deals with analyzing different types of data. The data typically used in data mining is in the format of a single table, with primitive datatypes as attributes. However, structured (complex) data, such as graphs, sequences, networks, text, image, multimedia and relational data, are receiving an increasing amount of interest in data mining. A major challenge is to treat and represent the mining of different types of structured data in a uniform fashion. A theoretical framework that unifies different data mining tasks, on different types of data can help to formalize the knowledge about the domain and provide a base for future research, unification and standardization. Next, automation and overall support of the Knowledge Discovery in Databases (KDD) process is also an important challenge in the domain of data mining. A formalization of the domain of data mining is a solution that addresses these challenges. It can directly support the development of a general framework for data mining, support the representation of the process of mining structured data, and allow the representation of the complete process of knowledge discovery.

In this thesis, we propose a reference modular ontology for the domain of data mining OntoDM, directly motivated by the need for formalization of the data mining domain. The OntoDM ontology is designed and implemented by following ontology best practices and design principles. Its distinguishing feature is that it uses Basic Formal Ontology (BFO) as an upper-level ontology and a template, a set of formally defined relations from Relational Ontology (RO) and other state-of-the-art ontologies, and reuses classes and relations from the Ontology of Biomedical Investigations (OBI), the Information Artifact Ontology (IAO), and the Software Ontology (SWO). This will ensure compatibility and connections with other ontologies and allow cross-domain reasoning capabilities. The OntoDM ontology is composed of three modules covering different aspects of data mining: OntoDT, which supports the representation of knowledge about datatypes and is based on an accepted ISO standard for datatypes in computer systems; OntoDM-core, which formalizes the key data mining entities for representing the mining of structured data in the context of a general framework for data mining; and OntoDM-KDD, which formalizes the knowledge discovery process based on the Cross Industry Standard Process for Data Mining (CRISP-DM) process model.

The OntoDT module provides a representation of the datatype entity, defines a taxonomy of datatype characterizing operations, and a taxonomy of datatype qualities. Furthermore, it defines a datatype taxonomy comprising classes and instances of primitive datatypes, generated datatypes (non-aggregate and aggregated datatypes), subtypes, and defined datatypes. With this structure, the module provides a generic mechanism for representing arbitrarily complex datatypes.

The OntoDM-core module formalizes the key data mining entities needed for the representation of mining structured data in the context of a general framework for data mining. These include the entities dataset, data mining task, generalization, data mining algorithm, and others. More specifically, it provides a representation of datasets, and a taxonomy of datasets based on the type of data. Next, it provides a representation of data mining tasks, and proposes a taxonomy of data mining tasks, predictive modeling tasks and hierarchi-

cal classification tasks. Furthermore, it provides a representation for generalizations, and proposes a taxonomy of generalizations and predictive models based on the types of data and generalization language. Moreover, it provides a representation of data mining algorithms, proposes a taxonomy of data mining algorithms, predictive modeling algorithms, and hierarchical classification algorithms, and generalizes the mechanism for representing data mining algorithms to represent general algorithms in computer science. In addition, the OntoDM-core module provides a representation of constraints and constraint-based data mining tasks and proposes a taxonomy thereof. Finally, the module provides a representation of data mining scenarios that includes data mining scenarios as a specification, data mining workflows, and the process of executing a data mining workflow.

The OntoDM-KDD module supports the representation of data mining investigations. It provides a representation of data mining investigation by directly extending classes from the OBI and IAO ontologies. Furthermore, it models each of the phases in a data mining investigation (such as application understanding, data understanding, data preparation, modeling, DM process evaluation, and deployment), and their inputs and outputs.

The OntoDM ontology and its three modules OntoDT, OntoDM-core, and OntoDM-KDD) were evaluated in order to assess their quality. The evaluation was performed by assessing the ontology against a set of design principles and best practices, and assessing whether the competency questions posed in the design phase were implemented in the language of the ontology. In addition, we provided a domain coverage assessment by comparing the OntoDM data mining tasks taxonomy with the data mining topic ontology constructed in a semi-automatic fashion from abstracts of articles from data mining conferences and journals.

The developed ontology supports a large variety of applications. We demonstrate the use and the application of the ontology by describing six use cases. The OntoDM ontology is used for the annotation of data mining algorithms; for the representation of data mining scenarios; for the annotation of data mining investigations; in cross domain applications to support ontology-based representation of QSAR modeling for drug discovery, as a mid-level ontology by the Expose ontology; and for the annotation of articles containing data mining terms in combination with text mining tools.

The novelties that the OntoDM ontology introduces and what distinguishes it from other related ontologies are the facts that it allows representation of mining of structured data and the general process of data mining in a principled way, it is based on a theoretical ontological framework and due to this it can be connected to other domain ontologies to support cross-domain applications. The OntoDM ontology is also the first ontology that supports the representation of the complete process of knowledge discovery.

In the future developments of the OntoDM ontology, we plan to focus on several aspects. First, we would like to align and map of our ontology to other upper-level ontologies. Second, we plan to extend the established ontological framework to represent entities about components of data mining algorithms, such as distance functions and kernel functions. Next, we plan to populate the ontology downward with instances. Furthermore, we plan to extend the representational framework for representing experiments for mining structured data in the context of experiment databases. Finally, we plan to include more contributors from the domain of data mining into the development of OntoDM and apply the OntoDM design principles to the development of ontologies for other areas of computer science.

Povzetek

Domena podatkovnega rudarjenja se ukvarja z analizo različnih vrst podatkov. Podatki, ki se običajno uporabljajo v podatkovnem rudarjenju, so po navadi v obliki ene same tabele, kjer vsaka vrstica vsebuje vrednost atributa, t.j., spremenljivke ki so primitivnega podatkovnega tipa. Obstaja vedno več zanimanjaVendar pa za strukturirane (kompleksne) podatke, kot so grafi, sekvence, mreže, besedila, slike, multimediski in relacijski podatki. Glavni izziv, ki se ga v tei disertaciji lotimo, je predstavitev in obravnava rudarjenja različnih vrst strukturiranih podatkov v enotni obliki. Teoretični okvir, ki združuje različne naloge podatkovnega rudarjenja različnih podatkovnih tipov, bi pomagal formalizirati znanje o domeni in s tem zagotoviti osnovo za nadaljnje raziskave, poenotenje in standardizacijo. Avtomatizacija in celotna podpora procesa odkrivanja znanja iz podatkovnih baz prav tako predstavljata pomemben izziv v domeni podatkovnega rudarjenja. Formalizacija osnovnih pojmov na področju podatkovnega rudarjenja bi omogocila predstavitev podatkovnega rudarjenja in reprezentacijo procesa podatkovnega rudarjenja strukturiranih podatkov ter obenem tudi procesov odkrivanja znanj iz podatkov.

V tej disertaciji predlagamo referenčno modularno ontologijo za domeno podatkovnega rudarjenja OntoDM, ki je neposredno utemeljena s potrebo po formalizaciji domene podatkovnega rudarjenja. Ontologija OntoDM je zasnovana in implementirana z upoštevanjem najučinkovitejših ontoloških praks in oblikovalskih načel. Uporablja temeljno formalno ontologijo (BFO) kot predlogo in kot ontologijo na višji ravni, množico formaliziranih relacij iz relacijske ontologije (RO) in drugih najrazvitejših ontologij; prav tako ponovno uporabi razrede in relacije iz ontologije biomedicinskih raziskav (OBI), ontologije informacijskih artefaktov (IAO) in ontologije programske opreme (SWO).

Ontologija OntoDM je sestavljena iz treh modulov, ki zajemajo različne vidike podatkovnega rudarjenja. OntoDT podpira reprezentacijo znanja o podatkovnih tipih in temelji na sprejetem ISO standardu za podatkovne tipe v računalniških sistemih. OntoDM-core formalizira najpomembnejše entitete podatkovnega rudarjenja za reprezentacijo rudarjenja strukturiranih podatkov v kontekstu teoretičnega okvira podatkovnega rudarjenja. OntoDM-KDD formalizira proces odkrivanja znanja in izhaja iz procesnega modela CRISP-DM (Cross Industry Standard Process for Data Mining).

Modul OntoDT omogoča predstavitev podatkovnih tipov, opredeljuje taksonomijo podatkovnih tipov, ki vključuje razrede in primere iz primitivnih podatkovnih tipov, generiranih podatkovnih tipov (nesestavljeni in sestavljeni podatkovni tipi), podtipov ter opredeljenih podatkovnih tipov. S takšno strukturo omogoča modul predstavitev poljubno kompleksnih podatkovnih tipov.

Modul OntoDM-core formalizira najpomembnejše entitete (v domeni) podatkovnega rudarjenja, ki so potrebne za predstavitev rudarjenja strukturiranih podatkov v kontekstu teoretičnega okvira podatkovnega rudarjenja: te vključujejo podatkovno množico, nalogo podatkovnega rudarjenja, modele, algoritm podatkovnega rudarjenja itd. Natančneje, modul omogoča predstavitev podatkovnih množic in taksonomijo podatkovnih množic, ki izhaja iz podatkovnega tipa. Prav tako omogoča predstavitev nalog podatkovnega rudarjenja ter predlaga taksonomijo nalog podatkovnega rudarjenja, taksonomijo nalog napovednega modeliranja, in taksonomijo nalog hierarhičnog razvrščanja. Poleg tega omogoča predstavitev

modelov ter predlaga taksonomijo splosnih modelov in napovednih modelov, ki temeljijo na podatkovnih tipih in jeziku modelov. Omogoča tudi reprezentacijo algoritmov podatkovnega rudarjenja in predlaga njihovo taksonomijo, ter taksonomijo algoritmov za napovedno modeliranje in taksonomijo algoritmov za hierarhično razvrščanje, prav tako posplošuje mehanizme predstavitev algoritmov podatkovnega rudarjenja v kontekstu predstavitve splošnih algoritmov v računalništvu. Modul OntoDM-core prav tako omogoča reprezentacijo omejitev in nalog podatkovnega rudarjenja, ki so opredeljene z omejitvami, pri čemer predlaga njihovo taksonomijo. Nenazadnje omogoča modul tudi predstavitev opisa sekvenc aktivnosti podatkovnega rudarjenja, ki vključuje specifikacijo, delotok podatkovnega rudarjenja ter proces izvedbe delotoka podatkovnega rudarjenja.

Modul OntoDM-KDD omogoca predstavitev procesa odkrivanja znanja s podatkovnim rudarjenjem. Modul omogoča pre poizvedbe podatkovnega rudarjenja z neposredno razširitvijo razredov iz OBI in IAO ontologij. Poleg tega modelira vsako fazo procesa odkrivanja znanja (denimo razumevanje uporabe, razumevanja podatkov, priprave podatkov, modeliranje, postopek vrednotenja podatkovnega rudarjenja in postavitev) ter njihovih vhodov in izhodov.

OntoDM ontologij ter njene pripadajoče tri module OntoDT, OntoDM-core in OntoDM-KDD smo ovrednotili z namenom opredelitve njihove kakovosti. Vrednotenje je potekalo na podlagi oblikovnih načel in najboljših praks, pri čemer se je ocenjevalo, ali lahko ontologija odgovori na kompetenčna vprašanja, ki so bila zastavljena v fazi oblikovanja. Poleg tega smo ocenili pokritost domene na podlagi primerjave taksonomije nalog podatkovnega rudarjenja s tematsko ontologijo podatkovnega rudarjenja, ki smo jo zgradili na polavtomatski način iz povzetkov člankov s konferenc o podatkovnem rudarjenju ter iz znanstvenih revij.

Na tak način razvita ontologija podpira široko paleto aplikacij. Njeno uporabo in možnosti aplikacije v ilustriramo na šestih opisanih primerih uporabe. OntoDM ontologij smo uporabili za označevanje algoritmov podatkovnega rudarjenja, za predstavitev scenarij aktivnosti podatkovnega rudarjenja ter za označitev procesa podatkovnega rudarjenja. V meddomenskih aplikacijah se OntoDM uporablja za podporo (na ontologiji temelječih) reprezentacij QSAR modeliranja za odkrivanje zdravil in kot srednjestopenjska ontologija (Exposé ontologija). OntoDM lahko uporabimo tudi ter za označevanje člankov, ki vsebujejo termine iz podatkovnega rudarjenja, v kombinaciji z orodji za rudarjenje besedil.

Novosti, ki jih uvaja ontologija OntoDM in jo obenem razlikujejo od ostalih sorodnih ontologij, so možnosti reprezentacije podatkovnega rudarjenja strukturiranih (kompleksnih) podatkov in generalnega procesa podatkovnega rudarjenja na načelni ravni, ki temelji na teoretičnem okviru, zaradi česar se lahko povezuje z ostalimi domenskimi ontologijami z namenom podpore meddomenskih aplikacij. Ontologija OntoDM je prav tako prva ontologija, ki omogoča reprezentacijo celotnega procesa odkrivanja znanja.

V okviru nadaljnjega razvoja OntoDM ontologije se nameravamo osredotočiti na nekaj vidikov. Najprej želimo poravnati in preslikati našo ontologijo na druge višjestopenjske ontologije. Prav tako načrtujemo razširitev predlaganega ontološkega okvira z namenom reprezentacije komponent algoritmov podatkovnega rudarjenja, kot sta funkcija razdalje in kernel funkcija. Nato nameravamo napolniti/dopolniti ontologijo s primeri. Poleg tega načrtujemo razširitev reprezentacijskega okvira z namenom reprezentacije poskusov/eksperimentov znotraj rudarjenja strukturiranih podatkov v kontekstu eksperimentalnih podatkovnih baz. Naposled bomo z namenom razvoja OntoDM vključili več sodelavcev s področja podatkovnega rudarjenji, pri čemer bomo uporabili načela oblikovanja OntoDM za razvoj ontologij z drugih področij računalništva.

Abbreviations

BFO	=	Basic Formal Ontology
CRISP-DM	=	Cross Industry Standard Process for Data Mining
CheTA	=	Chemistry using Text Annotations
CBDM	=	Constraint-based Data Mining
DM	=	Data Mining
DMO	=	Data Mining Ontologies
DMOP	=	Data Mining Optimization
DAG	=	Directed Acyclic Graph
DOLCE	=	Descriptive Ontology for Linguistic and Cognitive Engineering
DDI	=	Drug Discovery Investigations
EDM	=	Electric Discharge Machining
EXACT	=	Ontology of Experiment Actions
EXPO	=	Ontology of Scientific Experiments
FDL	=	Full Depth Labeling
GFO	=	General Formal Ontology
GDC	=	Generically Dependent Continuant
HC	=	Hierarchical Classification
HMC	=	Hierarchical Multi-label Classification
ICE	=	Information Content Entity
IAO	=	Information Artifact Ontology
ISO	=	International Organization for Standardization
KD	=	Knowledge Discovery
KDD	=	Knowledge Discovery in Databases
LABORS	=	Ontology of Automated Experimentation
MIREOT	=	Minimum Information to Reference an External Ontology Term
MPL	=	Multiple Paths Labeling
OBI	=	Ontology of Biomedical Investigations
OBO	=	Open Biomedical Ontologies
OSCAR	=	Open Source Chemistry Routines
PDL	=	Partial Depth Labeling
QSAR	=	Quantitative structure-activity relationship
RO	=	Relational Ontology
RDF	=	Resource Description Framework
SWO	=	Software Ontology
SVM	=	Support Vector Machines
SPL	=	Single Path Labeling
SDC	=	Specifically Dependent Continuant
SUMO	=	Suggested Upper Merged Ontology
URI	=	Universal Resource Identifier
YAMATO	=	Yet Another More Advanced Top-level Ontology
W3C	=	The World Wide Web Consortium

1 Introduction

In this chapter, we first present the background of the work presented in this thesis (Section 1.1). Next, we state the motivation for this work (Section 1.2). In addition, we present the goals of the work presented this thesis (Section 1.3). Furthermore, we list the original scientific contributions of this thesis (Section 1.4). We conclude this chapter with an overview of the thesis structure (Section 1.5).

1.1 Background

1.1.1 Challenges in the domain of data mining

Fayyad et al. (1996) define knowledge discovery in databases (KDD) as a "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". According to this definition, data mining (DM) is the central step in the KDD process concerned with the application of computational techniques (i.e., data mining algorithms implemented as computer programs) to find patterns in the data[1].

Data mining is concerned with analyzing different types of data. Besides data in the format of a single table (with primitive datatypes as attributes), most commonly used in data mining (Witten and Frank, 2005; Hand et al., 2001; Kaufman and Rousseeuw, 1990; Kononenko and Kukar, 2007; Han, 2005), structured (complex) data are receiving an increasing amount of interest (Bakir et al., 2007). These include graphs, sequences, networks, text, image, multimedia and relational data. Also, many data mining algorithms are designed to solve data mining tasks for specific types of data, most frequently defined for data represented in a single table. Examples of such tasks are classification, regression, or clustering. These tasks in essence can be defined on an arbitrary datatype and a theoretical framework that unifies different data mining tasks, on different types of data would help to formalize the knowledge about the domain and provide a base for future research, unification and standardization. A major challenge is to treat and represent the mining of different types of structured data in a uniform fashion. This was identified by Yang and Wu (2006) as one of the challenging problems in the domain. In addition, recent surveys of research challenges for data mining by Kriegel et al. (2007) and Dietterich et al. (2008) list the mining of complex data, the use of domain knowledge, and the support for complex knowledge discovery processes among the top-most open issues that have the best chance of providing the tools for building integrated artificial intelligence (AI) systems.

Džeroski (2007) addresses the ambitious task of formulating a general framework for data mining. In the paper, the author discusses the requirements that such a framework should fulfill. These include elegantly handling of different types of data, different data mining tasks, and different types of patterns and models. Furthermore, Džeroski discusses the design and implementation of data mining algorithms and their composition into scenarios for practical applications. In addition, the author develops his framework by laying some basic concepts, such as structured data, patterns and models, continues with data mining

[1] The patterns in this definition denote any kind of knowledge that is extracted in the process of data mining.

tasks and basic components of data mining algorithms, such as features, distances, kernels and refinement operators. Finally, these concepts are used to formulate constraint-based data mining task and the design of generic data mining algorithms.

Much of the research in recent years in the domain of KDD and DM has also focused on the automation and overall support of the KDD process, identified by Yang and Wu (2006) as another important challenge. This involves development of standards for performing the KDD process (such as for example the CRISP-DM methodology (Chapman et al., 1999)) as well as formal representations of the KDD processes in the form of workflows. In addition, specific issues addressed include methods that automate the composition of data mining operations into executable workflows. Finally, providing a mechanism for recording of results and the experimental settings of the data mining experiments obtained by executing the workflows on sets of data is becoming necessary for ensuring the repeatability and reuse of results.

On a Daghstul seminar entitled Data Mining: The Next Generation held in 2005, one of the discussion topics was the compositionality of data mining operators (Ramakrishnan et al., 2005). The participants of the seminar proposed to describe data mining operations in terms of their signatures, that is, in terms of the domain (inputs) and range (outputs) of the functions that they are computing. In order to structure the large number and variety of data processing and data mining operations, they proposed to organize the signatures into taxonomies. The purpose of taxonomies is to order the operators conceptually and also to get a better understanding of the commonalities and differences between them. At the higher levels of the taxonomy, the signatures are described by general terms, like patterns or models, and at the lower levels of hierarchy the signatures are specialized for certain types of patterns or models, such as for example trees and rules.

1.1.2 Formalization of scientific investigations

The term science, in general, refers both to scientific knowledge and the process of acquiring such knowledge. It includes any systematic field of study that relates to observed phenomena and involves claims that can be tested empirically. Džeroski et al. (2007) state two reasons why it is important to study the process of scientific discovery from a computational perspective. First, this allows us to understand how humans perform scientific discoveries. Second, this allows us to automate or provide computational support for different facets of science. The first step in the automation of science is to formally represent the process of science itself and its participants. For this purpose, one needs to use some knowledge representation structures.

Formalization of scientific investigations has proved to be a prominent area of research. This includes providing a formal representation of objects and processes involved in scientific investigations in some knowledge representation framework (Brachman and Levesque, 2004), such as terminologies, taxonomies, and ontologies. In recent years, the use of ontologies has become prominent in the area of computer science research and the application of computer science methods in management and representation of scientific and other kinds of information. In this sense, the term ontology has the meaning of a standardized terminological framework in terms of which the information is organized. Traditionally, ontology has been defined as the philosophical study of what exists: the study of kinds of entities in the reality, and the relationships that these entities bear to one another (Smith, 2003b).

Most prominent developments in the context of formal representation of scientific investigations have been done in biological and biomedical domains. Examples of formal representations of scientific investigations in the form of ontologies include ontology for The Robot Scientist project (King et al., 2009), where an ontology was developed to provide support for recording of experiments performed and designed by an autonomous robot for the purpose of formalizing the robot's research. Second example is the Ontology of Biomedical

Investigations (OBI), where the ontology provides a model for the design of an biological and clinical investigation, the protocols and instrumentation used, the materials used, the data generated and the type of analysis performed on it. The third example is the EXACT ontology (Soldatova et al., 2008), that formalizes biological laboratory protocols, and others.

The State-of-the-Art in formal representations of scientific investigations also includes initiatives to support and unify the representational mechanisms for representing scientific investigations under a single framework. Example of such initiative is the Open Biomedical Ontologies (OBO) Foundry (Smith et al., 2007). The OBO foundry includes involving developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of orthogonal inter-operable reference ontologies in the biomedical domain, such as the Gene Ontology (GO) (Ashburner et al., 2000), Chemical Entities of Biological Interest (ChEBI) Ontology (de Matos et al., 2010), and others.

1.1.3 Applied ontology

From the definition of ontology from the previous section, we can see that the problem that ontologies are addressing in general is focused on adopting a set of basic categories of objects, determining what (kinds of) entities fall within each of these categories of objects, and determining what relationships hold within and among different categories in the ontology. The ontological problem for computer science is identical to many of the problems in philosophical ontology and the success of constructing such an ontology is thus achievable by applying methods, insights and theories of philosophical ontology. The process of constructing an ontology means that we design a representational artifact that is intended to represent the universals and relations among universals that exist, either in a given domain of reality (e.g., the domain of data mining research) or across such domains. In this context, Smith et al. (2006) define an ontology as 'a representational artifact, comprising a taxonomy as a proper part, whose representational units are intended to designate some combination of universals, defined classes, and certain relations between them'.

Applied ontology is a branch of ontology science that involves practical applications of ontological resources to specific domains, such as biology, biomedicine, ecology, computer science, and others (Munn and Smith, 2008). In this context, a domain is a portion of the reality that forms the subject-matter of a single science or technology or mode of study (Smith et al., 2006). In addition, the ontology engineering of is a branch of ontology science that deals with the set of activities that concern the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tools and languages that support them. Ontology engineering is still a relatively new research field and some of the steps in ontology design remain manual and more of an art than craft. Recently, there has been significant progress in automatic ontology learning (Malaia, 2009), applications of text mining (Buitelaar and Cimiano, 2008), and ontology mapping (Lister et al., 2009). However, the construction of a high quality ontology with the use of automatic and even semi-automatic techniques still requires manual definition of the key upper level entities of the domain of interest.

There is a lack of literature covering the best practices regarding ontology design, its construction and maintenance. Simperl and Tempich (2006) provide a good reference list of the best practices observed for a set of ontologies they examine, that are currently relevant in ontology engineering. Furthermore, various collective ontology development efforts provide lists of additional principles that can be used for ontology development purposes. For example, the OBO Foundry effort provides a list of principles to which the member ontologies should adhere. The most common principle is to re-use of existing ontologies as much as possible and this is one of the major emphases of the OBO Foundry (Smith et al., 2007). Furthermore, Courtot et al. (2011) propose a set of guidelines to follow when reusing terms

and classes from external ontologies based on the same design principles. Finally, the most important ontology best practices are to use an upper-level ontology as template, to use a formal set of relations defined with respect to the upper-level ontology, reuse of the already developed ontologies as much as possible, and not allowing multiple inheritance (Soldatova and King, 2005).

An upper-level ontology is a high-level, domain independent ontology, providing a framework by which different systems can use a common knowledge base and from which more domain-specific ontologies can be derived. Examples of upper-level ontologies include SUMO (Niles and Pease, 2001), CyC (Lenat, 1995), DOLCE (Masolo et al., 2003), BFO (Grenon and Smith, 2004), GFO (Herre et al., 2006), and YAMATO (Mizoguchi, 2010). A domain ontology, on the other hand, specifies entities particular to a domain of interest and represents those entities and their relationships from a domain specific perspective. Reuse of well established ontologies in domain ontology development allows one to take an advantage of the semantic richness of relevant entities and logic built in the reused ontology.

Modern ontologies can get quite large and complex, which poses challenges to tools and users that are processing, editing, analyzing and reusing parts of the ontology. Consequently, the modularity of ontologies has been an active topic in ontology engineering. Modularization in its most generic meaning denotes the possibility to perceive a large knowledge repository (e.g., ontology) as a set of modules, that in some way are parts of and compose the whole (ontology) (Stuckenschmidt et al., 2009). In this context, modules are used to represent the knowledge of a sub-domain of interest.

Finally, the ultimate goal of every domain is to have a reference ontology representing the knowledge about the domain. In this sense, a reference ontology is a representational artifact that is analogous to a scientific theory, where the prime importance is the expressive completeness and adequacy to the facts of the reality. Additionally, reference ontologies usually represent heavy-weight ontologies are typically developed with much attention paid to the precise meaning of each entity, the organizing principles rooted in philosophy, semantically rigorous relations between entities, etc.

1.2 Motivation

The research presented in this thesis is motivated by the need to formalize the knowledge about the domain of data mining and knowledge discovery in a form of an ontology. First, as the area of data mining is developing rapidly, one of the most challenging problems deals with developing a general framework for mining of structured data and constraint-based data mining. By formalizing the knowledge about the data mining domain we take one step toward solving this problem. This would involve identifying the key entities in the domain and their relations (e.g., a dataset, a data example, a data mining task, a data mining algorithm etc). After the basic entities are identified and logically defined, we can build upon them and define more complex entities (e.g., a constraint, a constraint-based data mining task, a data mining query, a data mining scenario and a data mining experiment). In addition, including the ontology into some well established knowledge representation framework, such as the OBO foundry, would enable to link to other domains (application areas of DM) such as biology, ecology, and biomedicine. Furthermore, this would enable automated reasoning over the knowledge about the domain. Finally, formalization of the domain of data mining would directly support the development of a general framework for data mining and support the representation of the process of mining structured data.

Second, there exist several proposals for ontologies of data mining, but the majority of them are light-weight application oriented ontologies aimed at covering a particular use-case in data mining, are of a limited scope, and highly use-case dependent. Initial systems that include ontologies are used to describe systematically the processes in machine learning and data mining. These include the CAMLET system (Suyama et al., 1998), the Min-

ingMart system (Morik and Scholz, 2004), and the IDA system (Bernstein et al., 2005). Next, there are ontology developments to support workflow composition and planing of workflows (Bernstein et al., 2005; Žáková et al., 2010; Diamantini and Potena, 2008; Kietz et al., 2009), support data mining applications on the GRID (Cannataro and Comito, 2003; Brezany et al., 2007), meta-learning and meta-mining (Hilario et al., 2009), support machine learning experiments in the context of experiment databases (Vanschoren et al., 2012), support application domains (e.g., (Hardy et al., 2010)), and learning semantic representations from text (Peng et al., 2008; Melli, 2010; Vavpetič et al., 2011).

Some of the ontologies are parts of active projects, e.g, the DMOP ontology (Hilario et al., 2009), the DMWF ontology (Kietz et al., 2009), the KDDONTO ontology (Diamantini and Potena, 2008), and the Exposé ontology (Vanschoren et al., 2012). Regarding the ontological aspects, almost all of the proposed ontologies for data mining are not explicitly aligned to any upper-level ontology and do not use a set of formal community agreed (or any logically defined) relations. This leads to the non-compatibility of ontologies and to ontologies that can not be easily reused and extended. An exception is the Exposéontology. Regarding the coverage aspect, the proposed ontologies deal with mining data represented in a single table with primitive attributes as datatypes and mainly focus on the predictive data mining tasks, such as classification and regression. An exception is the KD Ontology (Žáková et al., 2010) which includes some very specific relational mining tasks and algorithms. In addition, the ontologies do not provide a flexible mechanism for extension and generalization toward representing data mining tasks and algorithms for mining structured data. Finally, data mining is a domain which needs a heavy-weight ontology with a broader scope, where much attention is paid to the precise meaning of each entity, semantically rigorous relations between entities and compliance to an upper-level ontology, and compatibility with ontologies for the domains of application (e.g., biology, environmental sciences).

Finally, an ontology of data mining should define what is the minimum information required for the description of a data mining investigation. Biology and biomedicine are leading the way in developing standards for recording and representation of scientific data and biological investigations (Taylor et al., 2008) (e.g., already more than 50 journals require compliance of the reporting in papers results of microarray experiments to the Minimum Information About a Microarray Experiment - MIAME standard (Brazma et al., 2001)). The researchers in the domain of data mining should follow this good practice and the ontology of data mining should support the development of standards for performing and recording of data mining investigations.

1.3 Goals

Our main goal in this thesis is to develop a reference modular heavy-weight domain ontology of data mining. First, in order to support representation of mining of structured data the ontology should contain a separate module for representing knowledge about datatypes and should be based on some commonly accepted standard for datatypes in computer systems (e.g., (International Organization for Standardization, 2007)). Next, the ontology should contain a separate module that formalizes the key data mining entities for representation of mining of structured data, based on the proposal for a general framework for DM (Džeroski, 2007). Finally, the ontology should contain a separate module that formalizes the knowledge discovery process and introduces data mining investigations based some commonly accepted methodology for performing data mining investigations (e.g. CRISP-DM methodology (Chapman et al., 1999)).

The ontology should be designed and implemented by following ontology best practices and design principles (e.g., (Simperl and Tempich, 2006; Smith et al., 2007; Courtot et al., 2011; Soldatova and King, 2005)). This includes using a upper-level ontology BFO (Basic Formal Ontology) (Grenon and Smith, 2004) as a template, using formally defined relations

from RO (Relational Ontology) (Smith et al., 2005), and reusing classes and relations from other ontologies for representing scientific investigations, such as OBI (Ontology of Biomedical Investigations) (Brinkman et al., 2010), IAO (Information Artifact Ontology)[2], EXACT (Ontology of Experiment Actions) (Soldatova et al., 2008), SWO (Software Ontology)[3] using them as mid-level ontologies.

Finally, having this as a baseline, the ontology should support a large variety of applications. These include applications to support annotations and the representation of data mining algorithms, data mining scenarios, and data mining investigations. Furthermore, the ontology should support cross domain applications and provide a link to application domains of data mining. For example, the ontology can provide support for the representation of Quantitative structure-activity relationship (QSAR) modeling process in the context of Drug Design Investigation. Finally, the ontology should be developed in a general fashion in order to be used as a mid-level and further extended by other ontologies, focusing on a specific part of the data mining domain, ontology reuse by others that use the same set of design principles.

1.4 Scientific contributions

The work presented in this thesis comprises several contributions to the area of data mining, knowledge discovery in databases, and applied ontology. Furthermore, parts of the work presented here are published in several publications, such as Panov et al. (2008), Panov et al. (2009), Panov et al. (2010), and Panov et al. (2012). The complete list of papers related to this thesis is given in Appendix 1. The main contributions of the work presented in this thesis summarized as follows:

1. *State-of-the-art literature survey on ontologies for representing scientific investigations and ontologies in the domain of data mining.*

2. *Design and implementation of a modular reference domain ontology of data mining (OntoDM) composed of three modules (OntoDT, OntoDM-core, and OntoDM-KDD).* The proposed ontology is designed using a set of ontology best practices and design principles, including the use of BFO as the upper-level ontology, RO as the ontology for relations, and heavy reuse of other ontologies for scientific investigations, such as IAO, OBI, SWO and others. In addition, we propose a three-level description structure for representing entities form the data mining domain, which includes the specification level, implementation level, and application level.

   (a) *Design and implementation of an ontology module for representing knowledge about datatypes OntoDT, based on the ISO 11404 standard for general purpose datatypes.* This module includes representations of datatype, datatype qualities and datatype characterizing operations. In addition, it establishes a generic mechanism for representing arbitrarily complex datatypes.

   (b) *Design and implementation of an ontology module for representing knowledge about core data mining entities for mining structured data OntoDM-core based on a proposal for a general framework for data mining.* This module includes representations of dataset, data mining tasks, generalizations, data mining algorithms, constraints, constraint-based data mining task, and data mining scenarios. In addition, it provides a generic structure for representing algorithms in computer science.

[2]IAO: `http://code.google.com/p/information-artifact-ontology/`
[3]SWO: `http://theswo.sourceforge.net/`

(c) *Design and implementation of an ontology module for representing knowledge about data mining investigations OntoDM-KDD, based on the CRISP-DM methodology.* This module includes representations of data mining investigations, the phases in a data mining investigations and the inputs and outputs of each phase of the investigation.

3. *Evaluation of the OntoDM ontology and its modules OntoDT, OntoDM-core and OntoDM-KDD modules.* The evaluation is performed in the context of design principles and competency questions. In addition, we provide a domain coverage evaluation using a data mining topic ontology.

4. *Demonstration of the use of the OntoDM ontology in different use cases.* This includes the use of the ontology for annotation of algorithms, representation of scenarios, annotation of data mining investigations, support for cross-domain applications, mid-level ontology used by other users, and annotation of data mining articles combined with text mining.

1.5 Thesis structure

This introductory chapter presented the background and the context of the thesis. It also provided the motivation for performing the research and the list of the main original scientific contributions. The rest of the thesis is organized as follows.

In Chapter 2, we present the background information about the work presented in the remainder of the thesis. We first present different definitions of ontologies in computer science and the view of ontology as a representational artifact. Next, we present different ontological architectures such as the upper-level, mid-level, and domain ontology. Furthermore, we present the ontology engineering process, the design of modular ontologies, and ontology support activities. Finally, we conclude this chapter with an overview of ontology languages, and ontology engineering tools.

In Chapter 3, we present the related work connected to this thesis. First, we present the state-of-the-art in ontologies for science such as upper-level ontology initiatives and candidates, ontologies for scientific experiments, and biomedical ontologies. Next, we focus on the state-of-the-art in formal representations of the domain of data mining and review the developments in the context of the GRID, data mining workflows, meta-learning (and meta-mining), machine learning experiments, application domains of data mining, and learning semantic representations from text. Finally, we conclude this chapter with other related approaches that include semantic data mining and knowledge discovery for ontology construction.

In Chapter 4, we present task of developing a reference modular heavy-weight ontology of data mining (OntoDM), its design and implementation. First, we present the background that led to the development of the OntoDM ontology. Next, we focus on the ontology design best practices and design principles to be used for the design of the ontology. Furthermore, we present the basic design of the OntoDM ontology by discussing the upper-level ontology used (BFO), mid-level ontologies used (IAO, OBI, SWO), and the set of relations. In addition, we present the OntoDM modular structure and a proposal for a three-level description structure (specification, implementation, application) for representing entities in the data mining domain. Finally, we conclude this chapter with the discussion of the implementation and maintenance issues for the OntoDM ontology.

In Chapter 5, we present the module for representing datatypes (OntoDT). First, we present the OntoDT module goal, scope and competency questions. Next, we focus on the implementation of the module in the specification and implementation description level. Finally, we conclude the chapter with the ontology module description, that includes rep-

resentation of datatype, characterizing operations, datatype qualities, and the taxonomy of datatypes (primitive datatypes, generated datatypes, subtypes, and defined datatypes).

In Chapter 6, we present the ontology module for representing core data mining entities for representing mining of structured data (OntoDM-core). First, we present the OntoDM-core module goal, scope and competency questions. Next, we focus on the implementation of the module at the specification, implementation and application description level. Finally, we conclude the chapter with the ontology module description, that includes the representation of dataset (data specification, dataset specification, and taxonomy of datasets), data mining task (definition of task based on data specification, taxonomy of fundamental tasks, taxonomy of predictive modeling tasks, and taxonomy of hierarchical classification tasks), generalizations (three-level description of generalizations, taxonomy of generalizations, and taxonomy of predictive models), data mining algorithm (three-level description of data mining algorithms, taxonomy of data mining algorithms, taxonomy of predictive modeling algorithms, taxonomy of hierarchical classification algorithms, and proposal for a generic structure for representing algorithms in computer science), constraints and constraint-based data mining task (taxonomy of constraints, and taxonomy of constraint-based data mining tasks), and data mining scenario (three-level description of data mining scenarios).

In Chapter 7, we present the ontology module for representing data mining investigations (OntoDM-KDD). First, we present the OntoDM-KDD module goal, scope and competencies. Next, we focus on the implementation of the module at the specification and application description level. Finally, we conclude the chapter with the ontology module description, that includes representation of data mining investigation, and the phases that compose an investigation such as application understanding, data understanding, data preparation, modeling, DM process evaluation, and deployment.

In Chapter 8, we present the ontology evaluation. First, we present the ontology metrics. Next, we focus on the ontology assessment in terms of the design principles and ontology best practices. Furthermore, we focus on ontology assessment toward the competency questions. Finally, we conclude this chapter with evaluation of the OntoDM domain coverage using a data mining topic ontology.

In Chapter 9, we present and discuss the selected use cases for the OntoDM ontology. First, we focus on the use of OntoDM for annotation of data mining algorithms with an example of annotation for the Clus software[4]. Next, we present how OntoDM can be used for representing scenarios by showing examples of two scenarios that are modeled with the ontology (a scenario for evaluating predictive models and a scenario for comparing algorithms). Furthermore, we present how OntoDM (more specifically OntoDM-KDD module) can be used for the annotation of data mining investigations on an example investigation presented in a journal article. Next, we present the Robot scientist use case and show how OntoDM can be used in combination with the DDI ontology to support ontology-based representation of QSAR modeling for drug discovery. Finally, we conclude this chapter with the use case showing how OntoDM can be used as a mid-level ontology and further extended by other ontologies and the use of OntoDM in combination with text mining techniques for annotating DM algorithms.

Finally, Chapter 10 provides the conclusions of the work presented, a summary of scientific contributions, and discussion of the future work.

[4]Clus: `http://dtai.cs.kuleuven.be/clus/`

2 Ontology

Aristotle (384 BC-322 BC), a Greek philosopher and a student of Plato, first defined ontology as the science of being as such in the Metaphysics (Aristotle and Warrington, 1956):

> "There is a science which studies 'Being qua Being', and the properties inherent in it in virtue of its own nature."

The notion of ontology can be considered from two different perspectives: one that originates from philosophy, and one that originates from computer science. The philosophical perspective focuses on categorical analysis and answers questions such as: 'What are the entities in the world?' and 'What are the categories of entities?'. In this context, ontology is traditionally defined "as the philosophical study of what exists: the study of the kinds of entities in reality, and the relationships that these entities bear to one another" (Smith, 2003a).

Recently, the term ontology has become popular in the area of computer and information sciences, and the application of computer science methods for the management of scientific information. In this context, the terminology has the meaning of a standardized terminological framework. The computer science perspective sees ontology as a technology, and focuses on the same set of questions as the philosophical perspective, but with a different intention. The idea of the use of ontology in computer science is to create engineering models of reality that can be used by software, directly interpreted and reasoned over by inference engines.

In this chapter, we first present different views and definitions of ontology in computer science (Section 2.1). Next, we present an overview of ontology architectures (Section 2.2). Furthermore, we present the objectives of ontological engineering, as a scientific discipline that deals with methods for constructing and maintenance of ontologies (Section 2.3). Finally, we present an overview of ontology languages and tools used to code, design, implement and manage ontologies (Section 2.4).

2.1 What is an ontology?

In recent years, the term 'ontology' has been used widely in the areas of computer and information sciences and applications of these areas to the management of scientific and other kinds of information. The ontological problem in computer science is to adopt a set of basic categories of objects, then to determine what kinds of entities fall into each of the categories, and finally determine the relationships that are valid within and among different categories of the ontology. The ontological problem in computer science can be successfully solved only if one applies the methods, insights and theories of philosophical ontologies.

In this section, we present different definitions of what an ontology is, given by different authors in the area of computer science (Section 2.1.1). Next, we present the view that an ontology is a representational artifact (Section 2.1.2). Finally, we briefly discuss the possible roles of an ontologies in computer science (Section 2.1.3).

2.1.1 Definitions of ontology in computer science

In computer science, one of the first definitions of ontology was given by Neches et al. (1991), who defines ontologies 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".

According to this definition, an ontology along with the explicitly defined terms includes also knowledge that can be inferred from it. Next, Gruber (1993) defined ontology as follows:

> "An ontology is an explicit specification of a conceptualization".

This definition tries to contrast the definition of ontology in philosophy by emphasizing that ontology is a product of computational engineering. The notion of 'conceptualization is defined by Genesereth and Nilsson (1987) as "objects, concepts, and other entities that are presumed to exist in some area of reality and the relations that hold among them".

Guarino and Giaretta (1995) analyze different ways of characterizing ontology by using seven definitions:

1. Ontology as a philosophical discipline.
2. Ontology as an informal conceptual system.
3. Ontology as a formal semantic account.
4. Ontology as a specification of a conceptualization.
5. Ontology as a representation of a conceptual system via a logical theory
   (a) characterized by specific formal properties.
   (b) characterized only by its specific purposes.
6. Ontology as the vocabulary used by a logical theory.
7. Ontology as a (meta-level) specification of a logical theory.

This set of definitions leads to the definition of ontology as: "a logical theory which gives an explicit, partial account of a conceptualization".

Guarino (1998) refines the definition by Guarino and Giaretta (1995) with the refinement that ontology also has to be:

> "A set of logical axioms designed to account for the intended meaning of a vocabulary".

Other definitions of ontology are based on the process of building the ontology. These definitions cover the relationships between ontologies and knowledge bases. For example, Bernaras et al. (1996) propose the following definition:

> "Ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base".

Another definition is given by Swartout et al. (1997) having in mind a design strategy to reuse large ontologies to create domain ontologies and knowledge bases:

> "An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base".

Mizoguchi et al. (1995) define ontology from a knowledge-based systems' point of view, as

> "A theory(system) of concepts/vocabulary used as building blocks of an information processing system".

Furthermore, Mizoguchi (2003) gives a compositional definition of ontology as follows:

> "An ontology consists of concepts, hierarchical (IS-A) organization of them, relations among them (in addition to IS-A and PART-OF), axioms to formalize the definitions and relations".

2.1.2 Ontology as a representational artifact

In this thesis, we use a set of definitions of terms used for and about ontology defined by Smith et al. (2006). In their article, the authors propose a reference terminology for ontology research and development in the biomedical domain. These include definitions of the following terms: entity, representation, representational artifact, universal, particular, relation, class, domain, and types of representational artifacts. In addition, Smith et al. (2006) list the basic types of representational artifacts that include terminologies, inventories, taxonomies, and ontologies. In the remainder of this section, we present an overview of these terms and definitions.

Entity, representation and representational artifact. *Entity* is defined as anything which exists, including objects, processes, qualities and states on all three levels (Smith et al., 2006). There is a distinction of three levels of entities wherever ontologies are used. *Level 1* entities include the objects, processes, qualities and states in the reality. *Level 2* entities include cognitive representations of the reality. Finally, *level 3* entities include concretizations of these cognitive representations in the form of representational artifacts.

A *representation* is for example an idea, image, record, or description that is about, or refers to, some entity or entities external to the representation. A *composite representation* is a representation built out of constituent sub-representations as their parts. The smallest constituent sub-representations are called *representational units*. A *cognitive representation* is a level 2 representation, whose representational units are ideas, thoughts, or beliefs in the mind of some cognitive subject. A *formalized representation* is a representation interpretable by a computer and published in a language with formal semantics.

A *representational artifact* is a level 3 representation, that is fixed in some medium in such a way that it can serve to make the cognitive representations existing in the minds of separate subjects publicly available in some enduring fashion. Examples of representational artifacts include: text, diagrams, maps, legends, controlled vocabularies, and others.

Particulars, universals and domain. The ontology engineer at level 1 is concerned with the identification of *particulars* (individuals or instances) in the reality. The particulars are divided into *continuants* and *occurents*: The former covers entities that exists in full at any time, and persist through time while maintaining their identity, while the latter covers entities that have temporal parts and that happen, unfold or develop through time. A *universal* is something that is shared in common by all these particulars which are its *instances*.

Some particulars receive *proper names* that can be used in representational artifacts. Particulars can be referred to also by using complex expressions that involve *general terms*. These can include:

- terms that join entities in the reality that share common characteristics in the reality which are exemplified (general terms of this group refer to universals);
- terms that join entities in the reality that share common characteristics and are not intrinsic to the entities in question; and

- terms that relate to specific collections of particulars connected to specific regions of space and time.

The term *portion of reality* denotes both single universals and particulars and their complex combinations. A *domain* is a portion of the reality that forms the subject-matter of a single science or technology or mode of study. Representational artifacts represent entities in domains constrained by the level of granularity. Entities bellow a given threshold value can be excluded from the domain because they are not significant to the specific scientific goals at hand.

Class and relation. A *collection of particulars* is a collection that has other particulars as its members. Usually we can use the same general terms to refer to both universals and collections of particulars. A *class* is a collection of all and only the particulars to which a given general term applies. In the cases where the general term refers to a universal, the corresponding class, called the *extension* of the universal, comprehends all and only those particulars which instantiate the corresponding universal.

Both particulars and universals are related to each other in various *relations*. Particulars stand in a relation of *instantiation* with the corresponding universals. This and other relations (e.g., part-hood, adjacency, derivation) can be divided into three groups: relations between particulars, relations between universals, and relations between universals and particulars. Through the use of relations, both classes and universals are organized in a trees, the former on the basis of the subclass relation and the latter on the basis of the IS-A relation.

Types of representational artifacts. Smith et al. (2006) give definitions of several types of representational artifacts. These include the following:

- a *terminology* is defined as "a representational artifact consisting of representational units which are the general terms of some natural language used to refer to entities in some specific domain";

- an *inventory* is defined as "a representational artifact built out of singular referring terms such as proper names or alphanumeric identifiers";

- a *taxonomy* is defined as "a tree-form graphical representational artifact with nodes representing universals or classes and edges representing IS-A or subset relations"; and

- an *ontology* is defined as "a representational artifact, comprising a taxonomy as a proper part, whose representational units are intended to designate some combination of universals, defined classes, and certain relations between them".

2.1.3 Roles of an ontology

Ontologies can have different roles in real life applications. They can be used as common vocabularies, as data structures, as tools that provide semantic interoperability, as tools for the systematization of knowledge, and as a meta-models of reality, and to provide a theory of content.

One of the fundamental roles of an ontology is to provide a common vocabulary. An ontology in a database is the conceptual schema. In this sense, an ontology provides the data structure appropriate for information description and exchange. Meta-data used in the semantic web are built on the basis of an ontology which constrains and partially defines the meaning of tags and values. Interpretation and translation of the meta-data can be done via ontologies. Ontologies thus play the role of glue which guarantees semantic interoperability among meta-data.

Knowledge systematization requires a well-established vocabulary of concepts in terms of which people describe phenomena, theories and target things under consideration. An ontology thus contributes to providing a backbone of systematization of knowledge. A model is usually built in the computer as an abstraction of the real-world target and the ontology provides us with concepts and relations among them which are used as building blocks of the model. Thus, an ontology specifies the models to build by giving guidelines and constraints which should be satisfied.

For example, in biology domains ontologies have been used for different purposes (Smith and Shah, 2007): as controlled vocabularies (Gene Ontology) (Ashburner et al., 2000), for representing encyclopedic knowledge (Foundation model of anatomy FMA) (Rosse and Mejino, 2003), as a specification of an information model (MAGE-OM, MAGE-ML, MGED ontology) (Ball and Brazma, 2006), for specification of a data interchange format (BioPax[1]), and representation of semantics of data for information integration (TAMBIS) (Ste et al., 2000).

2.2 Ontology architecture

Ontology architecture is the architecture used to structure the ontology. It addresses the levels of ontologies required, such as foundational, upper, middle, domain, sub-domain, application and reference ontology; and the mathematical, logical, and engineering constructs used to modularize and structure the ontological space. In this section, we first review ontologies at different levels, such as upper-level, mid-level and domain ontologies (Section 2.2.1). Next, we discuss the differences between an application and a reference ontology (Section 2.2.2). Finally, we discuss the differences between light-weight and heavy-weight ontologies (Section 2.2.3).

2.2.1 Ontology levels

An ontology architecture encompasses primarily three layers: upper ontologies, mid-level ontologies, and domain ontologies (see Figure 2.1) (Obrst, 2010). The upper-level and mid-level ontologies are also called foundational ontologies.

Upper-level ontology. An upper-level ontology is a high-level, domain independent ontology, providing a framework by which different systems can use a common knowledge base and from which more domain-specific ontologies can be derived. The entities in such an ontology are basic and universal. They provide expressiveness for a wide area of domains and ensure generality. An upper level ontology is limited to entities that are meta, generic, abstract and philosophical.

Upper-level ontologies specify the reality along two dimensions. In the first dimension, they state what are the basic categories of the reality. In the second dimension, they state what are the basic relations that hold within and among the objects that belong to the basic categories. The term ontological category in the case of upper-level ontologies can be interpreted as an universal that applies to every material domain of the reality. The ontological categories in an upper-level ontology should include all and only very basic and very general kinds of entities.

Mid-level ontology. A mid-level ontology serves as a bridge between general entities defined in the upper level ontology and the low-level domain specific entities defined in the domain ontology. The mid-level and upper level ontologies are intended to provide a mechanism for mapping of entities across domains easier. Mid-level ontologies may provide more concrete representation of abstract entities present in the upper-level ontologies.

[1]BioPax: `http://www.biopax.org/`

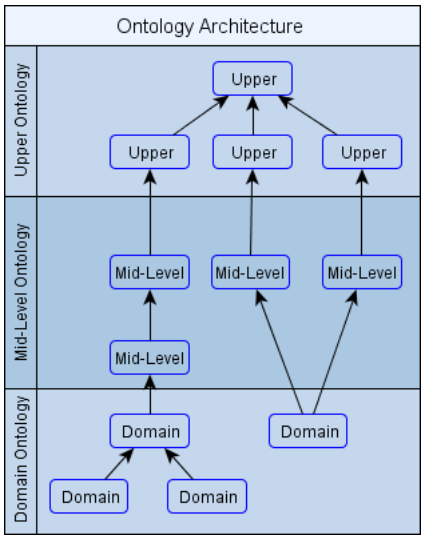


Figure 2.1: Upper, mid-level and domain ontologies.

Domain ontology. A domain ontology specifies entities particular to a domain of interest and represents these entities and their relationships from a domain specific perspective. Domain ontologies may be composed by importing mid-level ontologies. They may additionally extend the entities from the mid-level or upper level ontologies. Reuse of well established ontologies in domain ontology development allows one to take an advantage of the semantic richness of relevant entities and logic built in the reused ontology. Using common mid-level and upper-level ontologies is intended to ease the process of integrating and mapping of domain ontologies.

2.2.2 Reference and application ontologies

A reference ontology is a representational artifact that is analogous to a scientific theory, where the prime importance is the expressive completeness and adequacy to the facts of reality. Reference ontologies are constructed and structured based on objective relevance. An application ontology is a representational artifact designed to assist in the achievement of some specific goal. Application ontologies are constructed and structured in terms of goal-oriented relevance. They use and reuse portions of reference ontologies to accomplish their particular needs.

2.2.3 Light-weight and heavy-weight ontologies

Light-weight ontologies are usually taxonomies, where little attention is invested in rigorous definition of the entities, principles of entity organization, distinction between term and entity, etc. The main purpose of such a taxonomy can be, for example, to improve the performance of search engines, hence it is very use case dependent.

Heavy-weight ontologies are typically developed with much attention paid to the precise meaning of each entity, the organizing principles rooted in philosophy, semantically rigorous relations between entities, etc. Instance models are usually built based on those ontologies to model a target world, which requires careful modeling of the world to guarantee the consistency and realibility of the model. An upper-level ontology is a typical example of a heavy-weight ontology.

2.3 Ontology engineering

Ontological engineering refers to a set of activities that concern the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them. The engineering part of developing ontologies comprises a complex set of activities that are conducted during conceptualization, design, implementation and deployment of ontologies. There exist numerous ontology engineering methodologies, used for this purpose. The most popular ontology engineering methodologies include the Ushold and Kings' methodology (Uschold and King, 1995), the TOVE methodology (Fox and Grüninger, 1994), METHONTOLOGY (López et al., 1999), the On-To-Knowledge methodology (Sure et al., 2004), and others.

In this section, we first present the ontology design and development process (Section 2.3.1). Next, we present design principles for building modular ontologies (Section 2.3.2). Finally, we conclude this section with a brief overview of ontology development support activities (Section 2.3.3).

2.3.1 Ontology design and development

Ontology building is a complex process. It includes activities such as problem specification, domain knowledge acquisition and analysis, conceptual design and commitment to community ontologies, iterative construction and testing, publishing the ontology as a terminology, and populating a conforming knowledge base with ontology individuals.

In this section, we briefly discuss the main aspects of constructing an ontology. First, we discuss the determination of the domain and the extent of an ontology. Next, we focus on the process of information gathering. Further on, we discuss the two major design approaches in ontology engineering: top-down and bottom-up design. Finally, we discuss how to structure the domain information and implement the ontology.

Determining the domain and the extent of an ontology. The first step in ontology construction is to determine the domain of the ontology, by answering the question "what part of the reality does this ontology cover?". Providing the answer to this question allows us to focus the efforts to construct the ontology, from the start, by indicating what information needs to be included in the ontology and what information should be treated as "not-important" for the ontology of our domain of interest. An important distinction to have in mind when constructing an ontology is which entities go into the formal or upper-level ontology and which go to the domain ontology.

Relevance of upper ontologies. As we already discussed in Section 2.2.1, an upper-level ontology is a representation of the categories of objects and of the relationships within and among categories that are to be found in any domain of reality. A domain ontology, on the other hand, consists of a representation of the material categories, universals and relationships among universals that are to be found in some specific domain of the reality, such as genetics, anatomy, computer science, etc. The domain ontology has the feature that it will contain many categories, universals and relations that are not to be found in the formal ontology and other domain ontologies.

The formal ontologies (such as the upper-level and mid-level ontologies) are of a high relevance for the task of constructing domain ontologies. They are important in two aspects. First, the categories and relations in formal ontologies are well structured and defined. This allows us to organize the universals and relations in the domain of interest equally well by following and extending the categories in the formal ontology. Second, using formal ontologies to structure domain ontologies helps to ensure the interoperability between different domain ontologies. For these reasons it is important to consider what formal ontology categories

and relations are relevant for the domain at hand and take them into consideration when designing a domain ontology. Furthermore, it is very important to select a formal ontology with sufficient categories and relations that will allow it to handle the basic entities from the domain of interest.

The horizontal and vertical dimension of a domain ontology. There are two general dimensions along which the appropriate contents of a domain ontology is to be determined. The dimensions are: the horizontal, or relevance dimension, and the vertical, or granularity dimension.

The horizontal dimension of the content of the ontology determines what and how much existing information about a domain of interest should be included in the ontology. The ontology should include only the information that is relevant to the domain represented by the ontology. Two considerations should be kept in mind when deciding what is relevant to a given domain ontology. First, what is relevant to the ontological representation of a given domain will be determined by the reality itself. Second, what is relevant to an ontology will be determined by the purpose for which the ontology is being designed.

The vertical dimension of the content of the ontology determines the appropriate granularity of the ontology. In a way, this is one of the sub-problems of determining what is relevant to be represented in the ontology we are building. In other words, this means determining how fine-grained the ontology should be. It involves deciding the upper and the lower limits of entity size and complexity to be represented in the domain ontology being designed. In the case of reference ontologies, the focus of representation should be on the nature of the objects, and in the case of application ontologies the focus should be on the objectives set by the goal of the application.

Information gathering. Once the domain or scope of an ontology has been decided, what is needed is a systematic survey of the information necessary to represent the domain of interest. The result of the survey should answer the following (non-extensive) list of questions:

1. What are the domain universals and relations that need to be represented?
2. What are the best domain specific term or terms that should be used in representing the universals and relations?
3. What are the explicit upper and lower bounds of the ontology?
4. How much formal ontological apparatus is needed for dealing with the domain to be modeled?

Domain literature and domain experts. The best candidates for providing the state-of-the-art information about the domain of interest are the scientists that study the domain. The activity of developing a domain-specific ontology is usually interdisciplinary, and in the ideal case includes a number of competent domain experts. The goal of consulting domain literature and experts is to gain as thorough and systematic knowledge of the domain to be represented as possible. Furthermore, this allows us to determine what are the universals and relations between universals in the domain of interest. Finally this allows us to determine which invariant features and regularities in the domain play a central role in the scientific explanation of that domain.

Terminology and granularity. In addition to consultations with the domain experts and literature, and determining the domain universals and relations, one should also construct a domain terminology. A domain terminology is a set of terms characteristically used to talk

about the entities in the domain. The terminology should include for each term: a listing of common synonyms or alternative expressions; a clear natural language definition; and an initial statement of the most likely ontology category to which the entities referred to by the term should belong. The initial terminology can be constructed manually by consulting domain literature, or can be adopted from an existing thesauri or handbook. Finally, after consulting the domain experts and literature and the construction of a terminology, it will be possible to provide a rather precise statement of the levels of granularity represented by the ontology.

Amount of formal ontology apparatus. What formal ontological categories and relations are necessary for structuring a given domain ontology depends on the content of that domain ontology. The initial assessment of the issue of formal ontology categories and relations should take place at the phase of determining the scope of the ontology. A final determination should take place after a relatively thorough survey of domain specific information has taken place. This involves as a first step, the choice of an upper level ontology and a set of formal relations based on the upper level categories.

Top-down and bottom-up design. There exists two major design approaches for the use of an upper-level ontology: top-down and bottom up. In a top-down approach we use the upper-level ontology as the foundation to derive the entities in the domain ontology. In this case, the upper-level ontology provides a theoretical framework on which to build the domain ontology. In a bottom-up design approach, we perform a mapping of the domain ontology to the upper-level ontology. This approach is usually more challenging, as we can encounter inconsistencies between the domain and upper ontology.

The top-down design of ontology begins by determining the domain of the ontology. Next, one needs to gather the necessary information about what are the universals in the domain and what are the relations between universals. Furthermore, the gathered information is organized in the form of a representational artifact, such as a written document, or a diagram. Finally, the information in the representational artifact is analyzed in order to ensure: its logical, philosophical and scientific coherence; coherence and compatibility with other relevant ontologies; and human intelligibility.

Structuring of domain information and implementation. Once the domain information has been assembled; a terminology has been defined capturing the important domain universals and relations; and the appropriate scope, granularity and formal ontological apparatus have been determined; the next step is to structure the domain information in a systematic and coherent fashion. The goal of this process is to develop a representational artifact that is logically coherent, true, and unambiguous representation of the reality. This process of structuring the domain information should follow the best practices in ontology engineering. Finally, after structuring the domain information one needs to formalize the structured representational artifact in a computer tractable language, in order to implement the representational artifact in some specific computing context.

2.3.2 Design of modular ontologies

Modern ontologies can get quite large and complex, which poses challenges to tools and users that are processing, editing, analyzing and reusing parts of the ontology. Consequently, the modularity of ontologies has been an active topic in ontology engineering. Modularization in its most generic meaning denotes the possibility to perceive a large knowledge repository (e.g., ontology) as a set of modules, that in some way are parts of and compose the whole (ontology) (Stuckenschmidt et al., 2009). Modularization materializes the long-established complexity management technique known as divide and conquer.

In the case of modular ontologies, the efforts go into developing logically sensible modules, that is, parts of and ontology which offer strong logical guarantees for intuitive modular properties, such as coverage. This means that a module captures all the ontology's knowledge about a given sub-domain. A module in this sense is a subset of the ontology's axioms that provides coverage for a sub-domain, and each sub-domain determines such a module.

The ideas of the followers of the monolithic approach in ontology engineering is to introduce modularity in order to organize and manage domain coverage. An example is given by the OBO foundry initiative and their policy to relate OBO ontologies, which can be viewed as modules of the foundational ontology BFO. The subdivision of the overall ontology in OBO modules is a consequence of setting up the upper-level and the ontology relations, and the focus is on extending the system to cover the whole domain.

2.3.3 Ontology development support activities

Ontology development support activities include activities that help the development, maintenance and the use of ontologies for different purposes. Examples of such activities include: ontology learning, ontology merging, ontology mapping, ontology versioning, ontology transformation, and ontology evaluation. In this section, we briefly discuss each of them.

Ontology learning and integration. Ontology learning is a broad domain of research that consists of ontology enrichment, inconsistency resolution and ontology population. Ontology enrichment is the task of extending an existing ontology with additional concepts and relations and placing them in the correct position in the ontology. Inconsistency resolution is the task of resolving inconsistencies that appear in the ontology with the goal of producing a consistent ontology. Ontology population is the task of adding new instances of concepts into the ontology.

When one wants to use different ontologies together, the ontologies need to be combined in some way. This can be done by integrating the ontologies, which means that they are merged into a new ontology. An alternative method is ontology mapping, where the ontologies are kept separate. In both cases, the ontologies have to be aligned, which means that they need to be in mutual agreement.

Ontology merging and mapping. Ontology merging is the creation of an ontology from two or more source ontologies. The resulting ontology replaces the original source ontologies. Ontology alignment is the process of discovering similarities between two source ontologies. The input to the ontology alignment algorithm is a number of source ontologies and the output is a specification of correspondences between the ontologies.

Ontology mapping is an important step to achieve knowledge sharing and semantic integration in an environment in which knowledge and information have been represented with different ontologies. The process of ontology mapping specifies the semantic overlap between two ontologies.

Ontology versioning, transformation and evaluation. As changes to ontologies are necessary, it is very important to keep track of changes and establish a relation between successive revisions of one ontology and the relation between the ontology and its dependencies. The dependencies include: the data that conforms to the ontology, other ontologies that are built from or import the ontology, and the applications that use the ontology.

Ontology transformation consists of transcribing an ontology from one form to another. This includes its expression in another ontology language, or a reformulation into a restricted form of the same language, or with regard to a different vocabulary.

Ontology evaluation is the process of assessing a given ontology from the point of view of a particular criterion of application, to determine if it would suit a particular purpose.

2.4 Ontology languages and tools

One of the key decisions to take in the ontology development process is to select the language (or set of languages) in which the ontology will be implemented. There are a number of such languages for ontologies, both proprietary and standard-based. Recently, many ontology implementation languages have been created. Other general Knowledge Representation (KR) languages and systems have been also used for implementing ontologies, even though these have not been specifically created with this purpose. These include languages such as: RDF[2], OWL[3], OntoLingua (Farquhar et al., 1997), XBRL[4], the OBO language[5], and others.

Building ontologies is a complex and time consuming process. It can be even more so if ontology developers have to implement ontologies directly in an ontology language, without any kind of tool to support the process. To facilitate this task, the first ontology building environments (or ontology tools) were created in the mid-1990s. They provided interfaces that helped users carry out some of the main activities of the ontology development process, such as conceptualization, implementation, consistency checking, and documentation.

In this section, we give an overview of languages used to formally represent ontologies (Section 2.4.1), as well as an overview of tools for ontology engineering and management (Section 2.4.2).

2.4.1 Ontology languages

Ontology languages are formal languages used to construct ontologies. The requirements for ontology languages are as follows. The language has to have a well-defined syntax: this is a necessary condition for the machine-processing of ontologies. It needs to have a formal semantics, giving a precise description of the meaning of the sentences of an ontology: this allows us to support automated reasoning. Finally, the language has to have sufficient expressive power to model the domain of interest.

Ontology languages are usually declarative languages. They can be classified by their structure into: frame-based languages (e.g., F-Logic (Kifer et al., 1995)), description logic-based languages (e.g., OWL), and first-order logic-based languages (e.g., KIF[6]).

The Web Ontology Language (OWL) is a language developed by The World Wide Web Consortium (W3C[7]). OWL is designed as a common language for ontology representation and is based on DAML+OIL[8]. It is an extension of RDF Schema developed as a standard language for ontology representation that should enable the semantic web: hence, extensibility, modifiability and interoperability have been given the highest priority in the design of OWL. At the same time, OWL tries to achieve a good trade-off between scalability and expressive power.

2.4.2 Ontology engineering and management tools

There are numerous ontology engineering and management tools in use today. Most of them have resulted from efforts of research groups and university labs, so they are currently free for general use. The ontology tools are split into two large categories: specialized ontology engineering tools (Section 2.4.2.1) and integrated ontology engineering environments (Section 2.4.2.2).

[2]RDF:`http://www.w3.org/RDF/`
[3]OWL:`http://www.w3.org/TR/owl-features/`
[4]XBRL: `http://www.xbrl.org/`
[5]OBO foundry: `http://www.obofoundry.org/`
[6]KIF:`http://www-ksl.stanford.edu/knowledge-sharing/kif/`
[7]W3C:`http://www.w3.org/`
[8]DAML+OIL:`http://www.daml.org/language/`

2.4.2.1 Specialized ontology engineering tools

Specialized ontology engineering tools support a restricted set of activities of the entire ontology life-cycle (design, deployment, maintenance, evolution). They can be divided into ontology engineering tools, ontology combination tools, and ontology management tools.

An ontology engineering tool is any tool used for creating ontologies or similar semantic documents. From this perspective, these tools can be classified into ontology building tools (e.g., OilEd [9]) and ontology learning tools (e.g., Ontogen (Fortuna et al., 2007)).

When one wants to (r)euse different ontologies together, the ontologies have to be combined in some way. Tools that perform this task are called ontology combination tools. They include: ontology merging tools (e.g., OntoMerge[10]), ontology alignment tools (e.g., OLA[11]), ontology mapping tools (e.g., MAFRA [12]), ontology versioning tools (e.g., OntoView[13]), and ontology translation/transformation tools (e.g., OntoMorph [14]).

Software tools are available to support the different phases of ontology development. While ontology editors are useful during each step of the ontology building process, other types of ontology engineering tools, such as ontology management tools, are also needed along the way. These tools include the following: ontology evaluation tools (e.g., OntoGenerator), ontology storage and querying tools (e.g., JENA[15]), ontology-based annotation tools (e.g., OntoELAN (Chebotko et al., 2004)), and ontology visualization tools (e.g., WSMOViz (Kerrigan, 2006)).

2.4.2.2 Integrated ontology engineering environments

There exist integrated collections of specialized tools that can support (fully or partially) more than one activity of the ontology engineering life-cycle. These are called integrated ontology engineering environments. The need for these tools arises from the fact that the life cycle of ontology engineering is highly affected if the ontology is to be reused for building another ontology, or vice-versa. Examples of integrated ontology engineering environments include: KAON (Bozsak et al., 2002), Protégé (Noy et al., 2000), WebODE (Arpírez et al., 2003), OntoEdit (Sure et al., 2002), the NeOn Toolkit[16], the TopBraid Composer [17], OBO-Edit[18], and others.

Protégé (Noy et al., 2000) was developed by the Stanford Medical Informatics Department at the Stanford University School of Medicine. It was developed originally for use in the field of clinical medicine and the biomedical sciences, but it is now being used in many areas where the concepts can be modeled as a class hierarchy. It is a Java-based open-source tool for editing and managing ontologies. It is the most widely used domain-independent, freely available, platform-independent technology for developing and managing terminologies, ontologies, and knowledge bases in a broad range of application domains.

[9] OilEd:`http://oiled.man.ac.uk`
[10] OntoMerge:`http://cs-www.cs.yale.edu/homes/dvm/daml/ontology-translation.html`
[11] OLA:`http://ola.gforge.inria.fr/`
[12] MAFRA:`http://mafra-toolkit.sourceforge.net/`
[13] OntoView:`http://trac.biostr.washington.edu/trac/wiki/OntoView`
[14] OntoMorph:`http://www.isi.edu/~hans/ontomorph/presentation/ontomorph.html`
[15] JENA:`http://incubator.apache.org/jena/`
[16] NeOn Toolkit:`http://www.neon-toolkit.org`
[17] TopBraid Composer: `http://www.topquadrant.com/topbraidvomposer.html`
[18] OBO-Edit:`http://oboedit.org`

3 Related Work

In this chapter, we give an overview of existing work related to this dissertation. The related work falls into three major areas. First, we present the related work concerning with the development of ontologies to support scientific investigations (Section 3.1). Next, we present the state-of-the-art in formal representation of data mining entities (Section 3.2). Finally, we present other related approaches concerned with the use of domain ontologies in data mining and with learning of topic ontologies by using a knowledge discovery process (Section 3.3).

3.1 State-of-the-art in ontologies for science

The term science refers both to scientific knowledge and the process of acquiring such knowledge. It includes any systematic field of study that relates to observed phenomena and involves claims that can be tested empirically. Džeroski et al. (2007) state two reasons why it is important to study the process of scientific discovery from a computational perspective. First, this allows us to understand how humans perform scientific discoveries. Second, this allows us to automate or provide computational support for different facets of science. The first step in the automation of science is to formally represent the process of science itself and its participants. For this purpose, one needs to use knowledge structures such as ontologies and taxonomies.

In this section, we first present upper-level ontology initiatives and candidates used to represent scientific investigations in different domains (Section 3.1.1). Second, we present the existing ontologies for representing scientific experiments and automated experimentation (Section 3.1.2). Finally, we present the most influential ontologies and initiatives from the domain of biomedicine (Section 3.1.3).

3.1.1 Upper-level ontology initiatives and candidates

In this section, we present the basic characteristics and applications of several upper-level initiatives and candidates. These include the following: Suggested Upper Merged Ontology (SUMO), Upper Cyc Ontology (UCO), Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), Basic Formal Ontology (BFO), General Formal Ontology (GFO) , and Yet Another More Advanced Top-level Ontology (YAMATO).

Suggested Upper Merged Ontology (SUMO). The SUMO[1] upper-level ontology was created by merging several public ontologies into a single cohesive structure (Niles and Pease, 2001). These ontologies included the ones available on the Ontolingua server[2], Sowa's upper-level ontology (Sowa, 2000), and various mereotopological[3] theories. The ontology has been

[1] SUMO: `http://www.ontologyportal.org/`

[2] Ontolingua server: `http://www.ksl.stanford.edu/software/ontolingua/`

[3] Mereology is a philosophical discipline that deals with parts and the wholes they form. Mereotopology is a first-order theory that embodies mereological and topological concepts, of the relations among wholes, parts, parts of parts, and the boundaries between parts.

used for research and applications in search, linguistics and reasoning. SUMO core contains about 1000 classes and 4000 axioms. It is composed of the SUMO core ontology, the mid-level ontology (MILO) and a set of domain ontologies such as: communications, economy, finance, physical elements, and others.

CYC ontology. Cyc[4] provides a broad common-sense ontology, a higher-order language for defining and making assertions using the ontology, and an inference engine for reasoning over such a knowledge base (Lenat, 1995). The inference engine provides subclass reasoning, deduction according to rules, and numerous special reasoning modules. Portions of the Cyc ontology can be extracted and reused in other projects without the inclusion of the vast majority of the Cyc ontology. The Cyc corporation has placed the core Cyc ontology (OpenCyc) into the public domain so that it can be freely used. The Cyc ontology has been used and applied in the domains of natural language processing, word sense disambiguation, question answering, and others.

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE). The DOLCE[5] upper level ontology aims at capturing the ontological categories necessary for the representation of natural language and human commonsense (Masolo et al., 2003). The upper-level categories it introduces are thought of as cognitive artifacts, which ultimately depend on human perception, cultural imprints and social conventions. The categories include: abstract quality, abstract region, agentive physical object, amount of matter, physical quality, physical region, process, and others. The applications of the DOLCE ontology include multilingual information retrieval, web-based systems and services, and e-learning.

Basic Formal Ontology (BFO). The BFO ontology[6] is narrowly focused on the task of providing a genuine upper ontology which can be used in support of domain ontologies developed for scientific research, for example in biomedicine (Grenon and Smith, 2004). BFO recognizes a basic distinction between two kinds of entities: substantial entities (or continuants) and processual entities (or occurrents). Continuants, represent entities that endure through time, while maintaining their identity. Occurents represent entities that happen, unfold and develop in time. The characteristic feature of occurents, or processual entities, is that they are extended both in space and time. The BFO ontology is included within the framework of the OBO Foundry. Finally, BFO 2.0 is planned to be released soon and it also contains general domain independent relations from the Relational Ontology (RO) and the Ontology of Biomedical Investigations (OBI).

General Formal Ontology (GFO). GFO[7] is a foundational ontology integrating objects and processes (Herre et al., 2006). GFO has a three-layered meta-ontological architecture consisting of an abstract top-level, an abstract core level, and a basic level. The ontology includes objects (3D objects) as well as processes (4D entities), both integrated into one coherent framework. The ontology is designed to support interoperability by following the principles of ontological mapping and reduction. GFO is designed for applications, primarily in the medical, biological, and biomedical areas, but also in the fields of economics and sociology.

Yet Another More Advanced Top-level Ontology (YAMATO). The YAMATO[8] ontology (Mizoguchi, 2010) has been developed to address the deficiencies of other upper

[4]UCO: `http://www.cyc.com/cycdoc/vocab/vocab-toc.html`
[5]DOLCE: `http://www.loa-cnr.it/DOLCE.html`
[6]BFO: `http://www.ifomis.org/bfo/`
[7]GFO: `http://www.onto-med.de/ontologies/gfo/`
[8]YAMATO: `http://www.ei.sanken.osaka-u.ac.jp/hozo/onto_library/upperOnto.htm`

ontologies such as DOLCE, BFO, GFO, SUMO, CYC, etc. These include dealing with qualities and quantities, dealing with representation and content bearing things, and distinguishing processes and events. The current version of YAMATO has been extensively used in several projects such as the development of a medical ontology, an ontology of learning and instructional theories, ontology of genomics (GXO), modeling of mobile users' behavior, functional ontology, and others.

3.1.2 Scientific experiments

An experiment is a procedure carried out with the goal of verifying, falsifying, or establishing the validity of a hypothesis. Experiments vary greatly in their goal and scale, but always rely on repeatable procedure and logical analysis of the results. Therefore, a formal definition of experiments for analysis, annotation and sharing of results is a fundamental part of scientific practice. In this section, we review the EXPO ontology that represents a proposal for formalization of scientific experiments and an extension of this proposal, named the LABORS ontology, in the context of automatic experimentation.

Ontology of Scientific Experiments (EXPO). The generic ontology of experiments EXPO tries to define the principal entities necessary for the representation of scientific investigations (Soldatova and King, 2006). This includes general knowledge about scientific experimental design, methodology, and representation of results. EXPO defines types of investigations, such as computational investigation and physical investigation, and their principal components: investigator, method, result, and conclusion. The ontology uses a subset of SUMO as top classes, and a minimized set of relations in order to provide compliance with existing formalisms. The EXPO ontology is a valuable resource for describing experiments from various areas of research. This was demonstrated with the use of this ontology for the description of high-energy physics and phylogenetics experiments.

Ontology of Automated Experimentation (LABORS). The LABORS ontology is an extension of EXPO for the description of automated investigations for the Robot Scientist Project[9]. LABORS defines research units, such as investigation, study, test, trial and replicate: These are required for the description of complex multi-layered investigations carried out by a robot. LABORS is applied to produce experimental descriptions of the robot's investigations, resulting in a formalization of over 10,000 research units in a tree-like structure that is 10 levels deep. The formalization describes how the robot contributed to the discovery of new scientific knowledge (King et al., 2009).

3.1.3 Biomedical ontologies

Biomedical ontologies describe biological research or medical data, and one of the most important ontology languages, the OBO language, was designed specifically for the needs of biological research. Bio-ontologies are tools for annotation and data integration that provide researchers with a common vocabulary to describe and communicate results and give bio-informaticians tools for the functional analysis of microarray data, network modeling, semantic similarity for biological analysis and clinical diagnostics, as well as many other applications. In addition, in the domain of bioinformatics and biomedicine there exist efforts to standardize the methodology for information exchange and enabling construction of bioinformatics workflows. This involves specifying and representing information about datatypes.

In this subsection, we first present the OBO Foundry, and then proceed to describe ontologies from the bio domains that are relevant to this thesis, such as the RO ontology of

[9]Robot Scientist Project:`http://www.aber.ac.uk/compsci/Research/bio/robotsci/`

relations in biomedical domains, the OBI ontology, the IAO ontology, the SWO ontology, the EXACT ontology, the BioMoby project, and the EDAM ontology.

Open Biomedical Ontologies (OBO) The Open Biomedical Ontologies (OBO) project is an organization which hosts a library of ontologies for the biomedical domain (Smith et al., 2007). A wide variety of biomedical domains is covered by the OBO including ontologies of anatomy, development and disease for a number of key organisms; ontologies of biological sequence, function and process; and ontologies of biochemistry, cell types and behavior. To be included in OBO, ontologies need to conform to a set of design criteria which ensure their quality and inter-operability (see Section 4.1.2). The single most successful ontology of the OBO Foundry is the Gene Ontology (Ashburner et al., 2000), but many of the other ontologies that are members or aspiring members of the OBO Foundry are also widely used (Smith et al., 2007).

OBO Relation Ontology (RO). The OBO Relation Ontology (RO) is an ontology of the relationships that are used between entities in biomedical ontologies (Smith et al., 2005). The OBO-RO provides a set of basic relations and gives axioms for these. Among the relations provided in the OBO-RO are the IS-A relation, various mereotopological relations, participation, transformation, and derivation relations. For each relation, axioms specifying reflexivity, transitivity, and symmetry are provided. By providing these relations with consistent and unambiguous definitions, the RO aims to facilitate ontology inter-operability and to support advanced reasoning across these ontologies. New ontologies in the OBO foundry are required to comply with the OBO-RO. Recently, the OBO-RO was significantly modified and contains only relations specific for the biomedical domains, while all domain independent relations are now moved to BFO 2.0.

Ontology of Biomedical Investigations (OBI). The OBI ontology aims to provide a standard for the representation of biological and biomedical investigations. OBI is fully compliant with the existing formalisms in biomedical domains. OBI is an OBO Foundry candidate (Smith et al., 2007). The ontology supports consistent annotation of biomedical investigations regardless of the particular field of study (Brinkman et al., 2010). OBI defines an investigation as a process with several parts, including the planning of an overall study design, executing the designed study, and documenting the results. The OBI ontology employs rigid logic and semantics as it uses an upper-level ontology (BFO) and the RO relations to define the upper-level classes and a set of relations. OBI defines occurrences (processes) and continuants (materials, instruments, qualities, roles, functions) relevant to biomedical domains. The Data Transformation Branch is an OBI branch concerned with identifying and representing entities and relations needed to describe processes which produce output data given some input data: This branch is directly relevant to the work presented in this thesis.

Information Artifact Ontology (IAO). Due to the limitations of BFO in dealing with information, an Information Artifact Ontology (IAO)[10] has been recently proposed as a separate branch of the OBI project. The IAO ontology aims to be a mid-level ontology, dealing with information content entities (e.g., documents, file formats, specifications), processes that consume or produce information content entities (e.g., writing, documenting, measuring), material bearers of information (e.g., books, journals) and relations in which one of the relata is an information content entity (e.g., is-about, denotes, cites).

[10] `http://code.google.com/p/information-artifact-ontology/`

Software Ontology (SWO). The Software Ontology (SWO)[11] is a resource for describing bioinformatics software tools, their types, inputs, outputs, tasks, versions, provenance and data associated. SWO is part of the SWORD project (Software Ontology for Resource Description), an inter-disciplinary effort to capture software descriptions used in the preservation of data. The SWO uses a reduced version of the Basic Formal Ontology (BFO) upper ontology and subclasses and relations from the Experimental Factor Ontology (EFO), the Ontology of Biomedical Investigations (OBI) and the Information Artifact Ontology (IAO).

Ontology of Experiment Actions (EXACT). The ontology of experiment actions (EXACT) aims to provide a structured vocabulary of terms for the description of protocols in biomedical domains (Soldatova et al., 2008). The main contribution of this ontology is the formalization of biological laboratory protocols in order to enable the repeatability and reuse of already published experiment protocols. This ontology and the COW (Combining Ontologies with Workflows) software tool were employed in a use case to formalize laboratory protocols in the form of workflows (Maccagnan et al., 2010).

The BioMoby project. The BioMoby[12] project aimed to standardize the methodology for information exchange and access to analytical resources in bioinformatics, by developing the BioMoby Semantic Web Service specification (The BioMoby Consortium, 2008). The specification extends and modifies the core Web Service specification by defining: an ontology of data domains (Namespace Ontology), an ontology-based data representation syntax (Object Ontology), and an ontology of Web Service operational descriptions (Service Ontology). The Namespace Ontology is a flat controlled vocabulary and defines all valid data namespaces, that represent the underlying source of a data record. The purpose of the Object Ontology is to create a consistent machine-readable data syntax in order to provide a mechanism for automatic transformation of data formats and data integration. The Service Ontology is a simple subclass hierarchy that defines a set of data manipulation operations and types of data analysis.

EDAM ontology. The EMBRACE Data and Methods (EDAM)[13] is an ontology of general bioinformatics entities (Pettifer et al., 2010). EDAM only includes entities from the domain of bioinformatics, while it does not represent general computer science and biological entities. The ontology includes five sub-ontologies organized as simple hierarchies: topic, operation, data, format and identifier. The topic sub-ontology contains general bioinformatics subjects or categories, such as field of study, data, processing, analysis or technology. The operation sub-ontology contains the functions or processes performed by bioinformatics software tools. The data sub-ontology contains the types of data used in bioinformatics. The types of data represented in EDAM do not describe how the data is specified or represented and the types represented can be overlapping. The format sub-ontology contains the specific layout for encoding a specific type of data in computer files or memory. The identifier sub-ontology contains labels that identify data, resources or biological entities. The EDAM ontology has been used as background knowledge for the composition of bioinformatics workflows (Lamprecht et al., 2010).

[11] SWO: `http://theswo.sourceforge.net/`
[12] BioMoby: `http://biomoby.org/`
[13] EDAM: `http://edamontology.sourceforge.net/`

3.2 State-of-the-art in formal representations of data mining entities

The main developments in formal representation of data mining entities take place in the areas of data mining workflow construction, data mining services, and describing data mining resources on the GRID. Other research includes the formal representations of machine learning experiments in the context of experiment databases. Finally, there is an increasing interest in learning/discovering data mining entities from the data mining literature.

In this section, we first present the early systems that facilitate the semantic representations of data mining entities (Section 3.2.1). Next, we present the state-of-the-art in representations for the GRID (3.2.2). Furthermore, we give an overview of the representation of data mining entities in the context of data mining workflows (Section 3.2.3) and meta-mining (Section 3.2.4). Next, we focus of semantic representations of machine learning experiments (Section 3.2.5). Finally, we conclude with the data mining entities in application domains (Section 3.2.6) and learning semantic representations of data mining entities from text (Section 3.2.7).

3.2.1 Early work

Early work on the semantic representation of data mining entities includes attempts to describe systematically the processes in machine learning and data mining. These include the CAMLET system (Suyama et al., 1998), the MiningMart system (Morik and Scholz, 2004), and the IDA system (Bernstein et al., 2005). In the remainder of this section, we overview the semantic representations of data mining entities used in these systems. First, we focus on the CAMLET system that is used for the automatic composition of inductive learning systems. Next, we discuss the MiningMart system that uses a set of formalized descriptions of preprocessing methods in data mining to enable the reuse of data pre-processing best practices and store them in databases. Finally, we focus on the IDA system that provides users with systematic enumerations of valid sequences of data mining operators.

The CAMLET system. Suyama et al. (1998) proposed the CAMLET system that uses two primitive light-weight ontologies of machine learning entities to support the automatic composition of inductive learning systems: the process ontology and the object ontology. The process ontology represents the inductive learning methods that compose the inductive learning systems. The object ontology represents the data structures that are manipulated by the methods in the process ontology. The process ontology uses a limited set of customized relations in order to connect to the object ontology.

Mining Mart. Morik and Scholz (2004) proposed the MiningMart system for reusing best practices in data preprocessing. The system uses a light-weight ontology in the form of meta-data that represents the pre-processing entities (such as data and pre-processing chains) found in the use cases. The goal of the ontology is to automatically generate SQL code for the process of data preprocessing.

Intelligent Discovery Assistant. Bernstein et al. (2005) proposed a prototype of an Intelligent Discovery Assistant (IDA), which provides users with systematic enumerations of valid sequences of data mining operators (called data mining processes). Effective ranking of the processes by different criteria is also provided in order to facilitate the choice of data mining processes to execute in order to solve a specific data mining task. This automated system takes the advantage of an explicit ontology of data mining operators (algorithms). A light-weight ontology is used that contains only a hierarchy of data mining operators divided into three main classes: preprocessing operators, induction algorithms and post

processing operators. The leaves of the hierarchy are the actual operators. The ontology does not contain information about the internal structure of the operators: The taxonomy is produced only according to the role that the operator has in the knowledge discovery process.

3.2.2 GRID

The advancement of ontology languages and technologies toward the Semantic web enabled new developments in semantic representations of data mining entities for the purpose of distributed data mining and knowledge discovery on the GRID. In this section, we present the state-of-the-art in the context of the GRID. This includes the DAMON ontology and the GRIDMiner Assistant.

DAMON Ontology. In the context of GRID programming, Cannataro and Comito (2003) proposed a design and implementation of an ontology of data mining. The motivation for building the ontology came from the context of the author's work in the Knowledge GRID (Cannataro and Talia, 2003). The main goals of the ontology are to allow the semantic search of data mining software and other data mining resources and to assist the user by suggesting the software to use on the basis of the user's requirements and needs. The proposed DAMON (DAta Mining ONtology) ontology is built through a characterization of available data mining software.

GRIDMiner Assistant. Brezany et al. (2007) introduced an ontology-based framework for the automated construction of complex interactive data mining workflows as a means of improving productivity of GRID-enabled data systems. For this purpose they developed a data mining ontology which is based on concepts from industry standards such as: the predictive model mark-up language (PMML)[14], the cross industry standard process for data mining (CRISP-DM) (Chapman et al., 1999), WEKA (Witten and Frank, 2005), and the Java data mining API (Hornick et al., 2006).

3.2.3 Data mining workflows

In line with the research for constructing complex data mining workflows for data mining applications on the GRID, the problem of semi-automatic design of workflows for complex knowledge discovery tasks and planning of workflows has also been addressed by other researches. In this section, we focus on the KD Ontology, the KDDONTO ontology, and the DMWF ontology.

Knowledge Discovery (KD) Ontology. Žáková et al. (2008) proposed an ontology directly integrated with a planner in order to to automatically propose workflows for the given type of inputs and required outputs of the discovery process. The KD ontology formalizes the notions of a knowledge type and data mining algorithm. The planning algorithm accepts task descriptions expressed using the vocabulary of the ontology and produces abstract workflows as directed acyclic graphs, having in the nodes implemented algorithms. The ontology contains descriptions of propositional algorithms and algorithms for relational mining. The methodology was implemented in the Orange4WS Environment for service-oriented data mining (Žáková et al., 2009; Podpečan et al., 2011).

KDDONTO Ontology. Diamantini and Potena (2008) introduced a semantic based, service oriented framework for tools sharing and reuse, in order to give support for semantic enrichment (through semantic annotation of KDD tools) and deployment of tools as web

[14] `http://www.dmg.org/`

services. For describing the domain, they propose an ontology named KDDONTO (Diamantini et al., 2009) which is developed having in mind the central role of a KDD algorithm and its composition into data mining workflows. The ontology provides information required by the KDD composer to build valid workflows by introducing a similarity matching technique devised to recognize valid algorithmic sequences on the basis of their input/output pairs (Diamantini et al., 2010).

Data Mining Workflow (DMWF) Ontology. Kietz et al. (2009, 2010) presented a data mining ontology for workflow planning. The ontology is designed to contain all the information necessary to support a 3rd generation KDD Support System. This includes the objects manipulated by the system, the meta data needed, the operators (i.e., algorithms) used, and a goal description. The vocabulary of the ontology is used further for Hierarchical Task Network planning (HTN) to produce workflows. The workflow generator is tightly coupled with a meta-miner whose role is to rank the workflows and is based on the DMOP ontology (presented in more detail in Section 3.2.4).

3.2.4 Meta-learning and meta-mining: The DMOP ontology

Building upon the work of Bernstein et al. (2005), Kalousis et al. (2008) proposed an intelligent data mining assistant that combines planning and meta-learning for the automatic design of data mining workflows. Furthermore, Hilario et al. (2009) propose a data mining ontology designed to support meta-learning for algorithm and model selection in the context of data mining workflow optimization.

One of the main goals of the Data Mining Optimization (DMOP) Ontology is to support meta-mining (meta-analysis) of complete data mining experiments in order to extract workflow patterns (Hilario et al., 2011). The ontology contains representations of data mining tasks, algorithms, models, workflows and experiments, limited to the case of propositional data mining. In addition, the authors have developed a knowledge base by defining instances of the DMOP ontology. The DMOP ontology is not based on any upper-level ontology and uses a large set of customized special-purpose relations.

3.2.5 Machine learning experiments: The Exposé Ontology

As data mining and machine learning are experimental sciences, insight into the performance of a particular algorithm is obtained by implementing it and studying how it behaves on different datasets. Blockeel (2006) proposes an experimental methodology based on experiment databases in order to facilitate the repeatability of experiments and the generalizability of experimental results in machine learning. This was successfully implemented by Blockeel and Vanschoren (2007) in the ExpDB system running at KUL Leuven[15]. For this purpose, the authors created the Exposé ontology as a conceptual basis for the database (Vanschoren and Soldatova, 2010).

Vanschoren et al. (2008) propose an XML based language (named ExpML) for describing classification and regression experiments. In this process, the authors identified the main entities for formalizing a representation of machine learning experiments and implemented it in an ontology (named Exposé) (Vanschoren and Soldatova, 2010). The Exposé ontology describes machine learning experiments in a standardized fashion by representing the experiment context, evaluation metrics, performance estimation techniques, datasets and algorithms. The Exposé ontology is based on the same design principles and builds upon the work presented in this thesis as presented in more detail in Section 9.5.

[15] ExpDB:`http://expdb.cs.kuleuven.be/`

3.2.6 Data mining entities in application domains: The OpenTox ontology

In the context of their project OpenTox, Hardy et al. (2010) developed an algorithm ontology. The purpose of this ontology is to support the OpenTox Framework for predictive toxicology data management, algorithms, modeling, validation and reporting. In the OpenTox algorithm ontology, the algorithms are grouped in four major classes: descriptor calculation algorithms (focusing on the calculation of QSAR descriptors), data mining and learning algorithms, preprocessing algorithms, and utility algorithms.

3.2.7 Learning semantic representations from text

In this section, we present the state-of-the-art in learning semantic representations of data mining entities from large collections of text. First, we review a proposal for a descriptive framework for DM and KDD extracted from documents. Next, we focus on the KDDO1 ontology built from text corpus.

A descriptive framework. Peng et al. (2008) survey a large collection of data mining and knowledge discovery literature in order to identify and classify the data mining entities into high-level categories using a grounded theory approach and validating the classification using document clustering. As a result of the study, the authors have identified eight main areas of data mining and knowledge discovery: data mining tasks, learning methods and tasks, mining complex data, foundations of data mining, data mining software and systems, high-performance and distributed data mining, data mining applications, and data mining process/project. The identified areas were validated by fitting new data mining and knowledge discovery articles to the framework and by document clustering.

KDDO1 ontology. Melli (2010) presented a linguistically-grounded domain specific ontology for the KDD domain (named kddo1[16]) based on a semantic Wiki built upon paper abstracts accepted for the KDD-2009 conference. Each abstract has its concepts mentions identified and linked to the appropriate concept in the ontology.

3.3 Other related approaches

In this section, we review two related approaches. The first approach, semantic data mining focuses on the use of ontologies in the process of data mining (Section 3.3.1). The second approach, deals with the use of knowledge discovery methodology for ontology construction (Section 3.3.2).

3.3.1 Semantic data mining

The term semantic data mining denotes a new challenge of mining semantically annotated resources, with ontologies used as background knowledge by data mining approaches (which have semantic data). One possible scenario in the semantic data mining context is the use of domain ontologies as background knowledge (or input) for mining experimental data. This has been demonstrated in the case of semantic subgroup discovery (Lavrač et al., 2009) with the SEGS (Trajkovski et al., 2008) system in the area of micro-array data analysis by using as background knowledge three ontologies: Gene Ontology (GO[17]), KEGG[18] and ENTREZ[19]. Furthermore, the approach of Trajkovski et al. (2008) was generalized and implemented in

[16]kddo1: `http://www.gabormelli.com/Projects/kdd/data/kddo/kddo1/`
[17]GO:`http://www.geneontology.org/`
[18]KEGG:`http://www.genome.jp/kegg/`
[19]ENTREZ:`http://www.ncbi.nlm.nih.gov/sites/gquery`

the g-SEGS system that can exploit any OWL ontology as background knowledge (Vavpetič, 2011).

3.3.2 Knowledge discovery for ontology construction

Grobelnik and Mladenić (2006) propose a methodology for semi-automatic topic ontology construction analogous, to the CRISP-DM methodology (Chapman et al., 2000) defined for the knowledge discovery process. The proposed methodology consists of the following phases: domain understanding, data understanding, task definition, ontology learning, ontology evaluation and refinement with a human in the loop. The knowledge discovery techniques applicable in this context are text mining tools that can be applied to the task of ontology learning from text, such as Ontogen (Fortuna et al., 2007).

Data mining topic ontology. Vavpetič et al. (2011) present the semi-automatically constructed data mining topic ontology constructed from a large corpus of data mining papers, using a tool for semi-automatic ontology construction OntoGen (Fortuna et al., 2007). A topic ontology is a set of topics connected with different types of relations. The main purpose of the topic ontology is to assess the DMOP ontology developed in the e-LICO project. Additionally, the topic ontology was compared with the OntoDM ontology (presented in this thesis) and the KD ontology (Žáková et al., 2010).

4 OntoDM: Modular Ontology of Data Mining

In Chapter 1 of this thesis, we introduced the problem of formalization of the scientific domain of data mining. The task of formalizing the knowledge of a domain includes providing a formal representation of objects and processes involved in scientific investigations in some knowledge representation framework (Brachman and Levesque, 2004). In recent years, ontologies have become prominent in the area of computer science research and the application of computer science methods in the management and representation of scientific domains. The most substantial developments in terms of the formal representation of scientific investigations were achieved in the biological and biomedical domains (presented in Chapter 3).

The domain of data mining is developing rapidly. This poses a number challenges for the researchers working in the domain. One of the most challenging problems, as discussed in Chapter 1, deals with developing a general framework for data mining, which should treat the mining of structured data in a uniform fashion. In addition, much of the research in recent years, in the domain of KDD and DM, has focused on the automation of and overall support for the KDD process. This has been identified as another important challenge in the domain to be addressed by the research community.

By formalizing the knowledge about the data mining domain in the form of an ontology, we take one step forward toward addressing these challenges, as in the process we aim to identify the key entities in the domain and their relations (e.g., a dataset, a data example, a data mining task, a data mining algorithm etc). After the basic entities are identified and logically defined, we can build upon them and define more complex entities (e.g., a constraint, a constraint-based data mining task, a data mining query, a data mining scenario and a data mining experiment). Finally, the process of formalization should identify what is the minimum information required for the description and representation of a data mining investigation.

In Chapter 3, we presented and discussed several proposals for ontologies of data mining. The majority of them are light-weight application oriented ontologies aimed at covering a particular use-case in data mining. They are of a limited scope, and highly use-case dependent. However, the ultimate goal of every domain is to have a reference ontology representing the general knowledge about the domain. In this thesis, we address the task of developing a reference modular domain ontology of data mining named OntoDM. The ontology is designed and implemented by following ontology best practices and design principles (Simperl and Tempich, 2006; Smith et al., 2007; Courtot et al., 2011; Soldatova and King, 2005). This includes using upper-level ontology BFO (Basic Formal Ontology) (Grenon and Smith, 2004) as a template, using formaly defined relations from RO (Relational Ontology) (Smith et al., 2005), and reusing classes and relations from other ontologies for representing scientific investigations, such as OBI (Ontology of Biomedical Investigations) (Brinkman et al., 2010), IAO (Information Artifact Ontology)[1], EXACT (Ontology of Experiment Actions) (Soldatova et al., 2008), SWO (Software Ontology)[2], using them as mid-level ontologies.

[1]IAO: `http://code.google.com/p/information-artifact-ontology/`
[2]SWO: `http://theswo.sourceforge.net/`

In this chapter, we first present the ontology design best practices and design principles (Section 4.1). Next, we present the basic design of the OntoDM ontology (Section 4.2). Next, we focus on the implementation and maintenance issues of the proposed ontology (Section 4.3). Finally, we conclude this chapter with a proposal for a three-level description structure for representing entities in the data mining domain based on the design principles (Section 4.4).

4.1 Ontology design best practices and design principles

There is a relative lack of literature covering the best practices regarding ontologies, their construction and maintenance. Simperl and Tempich (2006) examine 34 ontologies, their development process, cost and methodology employed. They provide a good reference list of best practices observed for these ontologies, currently relevant in ontology engineering. Various collective ontology development efforts provide lists of additional principles, which can be used for ontology development purposes. For example, the OBO Foundry effort provides a list of principles to which the member ontologies should adhere. The most common principle is to re-use of existing ontologies as much as possible and this is one of the major emphases of the OBO Foundry (Smith et al., 2007). Furthermore, Courtot et al. (2011) propose a set of guidelines to follow when reusing terms and classes from external ontologies based on same design principles. Finally, the NeOn[3] methodology also suggests guidelines for constructing individual ontologies by reusing and re-engineering other domain ontologies.

In this section, we summarize the design principles and ontology best practices, which we follow in addressing the task of constructing modular ontology for data mining. First, we give an overview of ontology best practices from different view points (Section 4.1.1). Next, we present the OBO Foundry principles that we use in designing of ontology for data mining (Section 4.1.2). Finally, we describe the MIREOT principle for reusing terms and classes from external ontologies (Section 4.1.3).

4.1.1 Summary of ontology best practices

Ontology best practices refer to how the ontology is scoped, designed and constructed. In this subsection, we summarize the general best practices used in the design and implementation phase of constructing our ontology for data mining. The best practices focus on engineering approaches, which need to be taken into account when deciding on the scope and content of the ontology, the ontology design and structure, naming and vocabulary, documentation, collaboration, and testing of the ontology.

Scope and content. One of the most important best practices in ontology engineering, in the context of deciding on the ontology scope and content, is to provide a balanced coverage of the subject domain. This is done by providing uniform breadth and depth of coverage in the ontology across its scope. The goal should be to embed the domain coverage into a proper context by providing the ability to inter-operate with other ontologies: This presents a major strength of an ontology. Next, one should reuse the structure of already developed state-of-the-art ontologies as much as possible. Furthermore, there should be clear and precisely defined relations, that will ensure that the relationships between classes in the ontology are coherent. Finally, one should always rely upon a set of core ontologies for external re-use purposes and map to external ontologies, in order to increase the likelihood of sharing and interoperability.

[3]Neon Project:`http://www.neon-project.org/`

Design and structure. In the context of designing a domain ontology, one of the first steps the ontology engineer needs to take is to start with a light-weight ontology. This is done in order to identify the entities in the domain and their relationships, and represents an intermediate step towards constructing a heavy-weight ontology of a domain of interest. This approach allows incrementally built ontologies. Furthermore, the ontologies should be built in a modular fashion and the domain and problem space should be split into logical clusters. Next, the ontology should be implemented in a machine-processable language, such as the OWL or RDF Schema. Finally, the ontology engineer needs to use annotation properties, as much as possible, to promote the usefulness and human readability of the ontology.

In the context of structuring a domain ontology, the ontology engineer should always try to structurally split domain classes (ABox) from instances (TBox). This enables easier maintenance, better understandability of the ontology, and scalability. Next, domains and ranges should be assigned to the used relations, in order to assist in testing and to ensure the coherence of the domain ontology. Furthermore, relation restrictions should be assigned with care. Finally, the use of disjoint classes is preferred in order to separate classes from one another. If not explicitly stated, the child classes from the same parent are assumed to overlap.

Naming and vocabulary. For entity naming, there are naming conventions, which are used in the ontology developing communities, such as biomedical ontology communities. The conventions suggest that all classes in the ontology should be named as single nouns. Next, all relations should be named as verb sense, e.g., hasProperty. For related relations or classes one should use common and descriptive prefixes and suffixes. Finally, where applicable, the inverse relations should be defined.

All classes and relations in an ontology should be defined. Providing clear definitions, along with the coherency of its structure, gives an ontology its semantics. Next, the ontology engineer should provide a preferred label annotation property that is used for human readable purposes and in user interfaces. Furthermore, one should assign logical and short names to ontology namespaces, such as name:IDXXX, with a preferred maximum of five letters. Finally, if needed, multi-lingual capabilities should be enabled in all definitions and labels.

Documentation, collaboration and testing. The first requirement of documentation best practice is a properly constructed and commented ontology. The entire ontology vocabulary should be documented via a dedicated system that allows finding, selecting and editing of ontology terms. Supplementary documentation should be provided for ontology construction and maintenance, including naming, selection, completeness and other criteria. If possible one should provide a complete Wiki-like documentation system for use cases, best practices, evaluation and testing metrics, tools installation and use, and all aspects of the ontology life-cycle. Finally, a complete graph of the ontology should be made available via graph visualization tools to aid understanding of the ontology.

Collaboration is an implementation best practice. The first focus of collaboration best practices is on the re-use of already agreed-up structures and vocabularies. The second focus is on the use of improved processes for consensus making, including tools support, to enable work groups to identify and decide upon the terminology, definitions, alternative labels, and relations between the classes.

Testing best practices are very important in the implementation and deployment phases of the ontology by invoking reasoners that can detect inconsistencies. Next, we can test new relations, aided by invoking reasoners, which will identify inconsistencies. Furthermore, we can use external knowledge bases and ontologies to conduct coherency testing for the basic structure and relationships in the ontology. Finally, we need to evolve the ontology

specification to include necessary and sufficient conditions to enable more complete reasoner testing for consistency and coherence.

4.1.2 OBO Foundry principles

The OBO Foundry[4] (Smith et al., 2007) is a collaborative experiment involving developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of orthogonal inter-operable reference ontologies in the biomedical domain. It is an attempt to apply the scientific method to the task of ontology development, where the scientific method rests on constant criticism and on the assumption that no resource will ever exist in a form in which it cannot be further improved. By joining the initiative, the authors of an ontology commit to its maintenance in light of scientific advances and for working with other members of the community to ensure the improvement of these principles over time.

The OBO foundry has released a set of accepted OBO principles and a set of OBO principles that are under discussion, both presented in Table 4.1. Currently, there are 13 accepted and 6 principles that are under discussion. The accepted principles include the principles such as openness, use of a common formal language, use of unique identifiers (URIs), use of versioning systems for tracking changes, providing textual definitions, use of relations from the RO, documentation of the ontology, use of naming conventions, and others. The principles that are under discussion include principles such as providing all definitions in genus-differentia form[5], extending downward some fragment of BFO upper-level ontology, contain only classes with single inheritance, and others.

4.1.3 MIREOT guidelines

The Minimum Information to Reference an External Ontology Term (MIREOT) guidelines were created to help the development of the Ontology of Biomedical Investigations (Courtot et al., 2011). The guidelines provide a mechanism for reusing existing ontology resources, and therefore avoiding duplication of efforts and ensuring orthogonality. This set of guidelines focuses on the task of importing single external classes from other ontologies into an ontology that is currently under development. The process of importing external classes consists of first gathering the minimum information for the external class and then using this minimum information to fetch additional annotation elements from the ontology, such as labels and definitions.

The minimum information needed to reference an external class is the source ontology URI (Universal Resource Identifier) and the external term's URI. These items are usually stable and can be used unambiguously to reference the external class from the target ontology. The minimum information needed to integrate the class in the new ontology is its position in the hierarchy, more specifically the URI of its direct subclass in the target ontology. To summarize, the information needed to consistently reference and reuse an external class is the source ontology URI, the source class URI and the target direct superclass URI.

After establishing a set of basic guidelines for referencing an external ontology class, a web based system named OntoFox[6] (Xiang et al., 2010) was developed in order to assist the end users in extracting classes from source ontologies and reuse them in their ontologies. OntoFox additionally allows easily configurable options for selecting annotation properties

[4]OBO Foundry: `http://www.obofoundry.org/`

[5]A genus-differentia definition is a type of intensional definition which defines a species as a subtype of a genus satisfying certain conditions (the differentia). Thus, the definiendum in such definitions is always a species (and not an individual), while the definiens consists of two parts: a genus (or family) which is a pre-defined term that includes the species defined as a subtype, and the differentia which is the condition that distinguishes the species from other instances of the same genus.

[6]OntoFox:`http://ontofox.hegroup.org/`

Table 4.1: OBO Foundry principles.

Accepted OBO Principles		
ID	**Name**	**Description**
FP 001	open	The ontology must be open and available to be used by all without any constraint.
FP 002	format	The ontology is in, or can be expressed in, a common formal language with accepted syntax, such as for example OBO Format or OWL.
FP 003	URIs	Each class and relation in the ontology must have a unique URI identifier.
FP 004	versioning	The ontology developers have procedures for identifying distinct successive versions, by using metadata for tracking the changes.
FP 005	delineated content	The ontology has a clearly specified and clearly delineated content.
FP 006	textual definitions	The ontology includes textual definitions for a substantial and representative fraction of the terms, plus equivalent formal definitions for a substantial number of terms.
FP 007	relations	The ontology uses relations which are unambiguously defined following the pattern of definitions laid down in OBO Relational Ontology.
FP 008	documented	The ontology is well documented (e.g., in a published paper or in a manual for developers and users).
FP 009	users	The ontology has a set of independent users (documented, e.g., via pointers in external URIs, use in cross-products)
FP 010	collaboration	The ontology will be developed collaboratively.
FP 011	locus of authority	There should be a single person who is responsible for the ontology, ensuring continued maintenance in light of scientific advance and response to user feedback.
FP 012	naming conventions	The ontology should follow a naming convention.
FP 016	maintenance	The authors of the ontology commit to its maintenance in light of scientific advance.
OBO Principles under discussion		
ID	**Name**	**Description**
FP 013	genus differentia	All ontology definitions are in genus-differentia form.
FP 014	BFO	Ontologies should be conceivable as the result of populating downward some fragment of BFO 2.0.
FP 015	single inheritance	The ontology should be conceived as consisting of a core of asserted single inheritance links, with further IS-A relations inferred.
FP 017	instantability	All the types represented by the terms in the ontology should be instantiable.
FP 018	orthogonality	For each domain, there should be convergence upon a single ontology that is recommended for use by those who want to become involved with the Foundry initiative.
FP 019	content	The ontology must be a faithful representation of the domain and fit for the stated purpose.

and inclusion of all or a computed subset of classes between the low and top level classes. The output of the system is given in OWL format and can be directly imported into a developer's ontology. The system currently supports a large selection of ontologies from the OBO library, such as the OBI ontology, the IAO ontology, the GO ontology and others.

4.2 The basic design of OntoDM

In the design of OntoDM ontology, we take into consideration the best practices in ontology engineering outlined above. In Figure 4.1 we present the structure of the OntoDM ontology. We use the upper level ontology BFO (Basic Formal Ontology) version 1.1[7] as a template to define the upper level classes (see Section 3.1.1). Furthermore, we use the OBO Relational Ontology (RO)[8] (Smith et al., 2005) and an extended set of RO relations to define the semantics of the relationships between the data mining entities. At the mid-level, we re-use and extend existing ontological resources where possible, such as OBO Foundry[9] candidate ontologies, the OBI ontology and the IAO ontology, and other ontologies used for representing of scientific investigations. Finally, the OntoDM ontology is structured as a modular ontology composing of three modules: OntoDT, OntoDM-core, and OntoDM-KDD.

By establishing these design decisions we achieve *is-a* completeness and single *is-a* inheritance for all classes. BFO an RO were chosen as they are widely accepted, especially within OBO ontologies. The use of an upper level ontology and relations based on upper level classes eases the interoperability of the classes with other external ontologies based on same foundational principles (such as the DDI ontology (Qi et al., 2010), and the EXPOSE ontology (Vanschoren and Soldatova, 2010)). The re-use of classes from other ontologies enables cross-domain reasoning and avoids duplication of efforts. In addition, the OntoDM ontology aims to follow the OBO Foundry (Smith et al., 2007) principles in ontology engineering that are widely accepted in the biomedical domain (see Section 4.1.2). Consequently, the ontology is developed to be complementary to and integrated with state-of-the-art ontologies for representing scientific investigations.

In this section, we first analyze and present the upper-level classes imported from the BFO upper-level ontology that are in context of the OntoDM ontology (Section 4.2.1). Next, we present the external classes that are reused and extended by our data mining ontology (Section 4.2.2). Next, we focus on the set of relations used in our data mining ontology (Section 4.2.3). Finally, we present the modular structure of the OntoDM ontology (Section 4.2.4).

4.2.1 Upper-level classes: The BFO upper-level ontology

The BFO upper-level ontology is primarily focused on the task of providing a genuine upper ontology which can be used to support domain ontologies developed for scientific research (Grenon and Smith, 2004). The most prominent use of the BFO upper ontology is in the area of biomedicine. BFO recognizes a basic distinction between two kinds of entities: substantial entities or continuants and processual entities or occurrents.

In Figure 4.2, we present a part of the BFO v1.1 class hierarchy. The violet boxes represent the classes that are reused and extended further in the OntoDM ontology proposed in this thesis. In the remainder of this section, we present the basic definitions of the BFO classes that are reused and extend for the purpose of representing data mining entities: the continuant classes (Section 4.2.1.1) and the occurent classes (Section 4.2.1.2).

[7]BFO: `http://www.ifomis.org/bfo`
[8]RO: `http://purl.org/obo/owl/OBO_REL`
[9]OBO Foundry: `http://ontoworld.org/wiki/OBO_foundry`

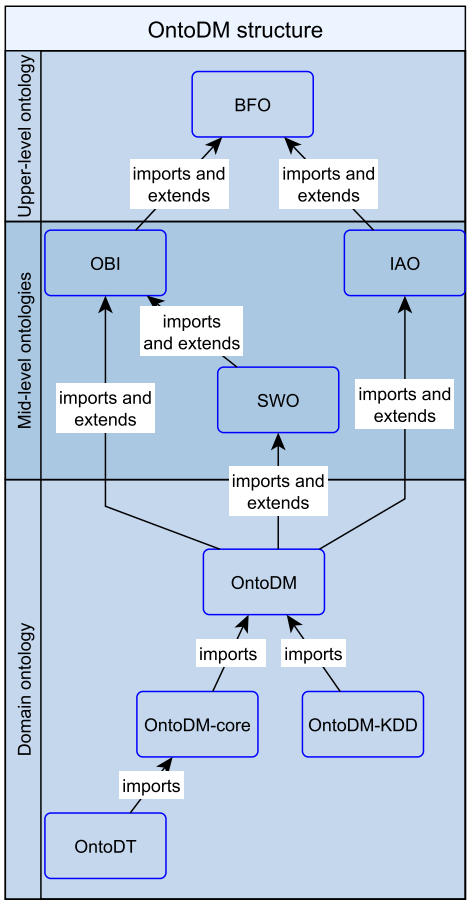


Figure 4.1: OntoDM modules.

4.2.1.1 Continuants

In the BFO ontology, a *continuant* is defined as an entity that exists in full at any time in which it exists at all, persists through time while maintaining its identity, and has no temporal parts (e.g., dataset, data mining task). The BFO ontology distinguishes between three types of continuants: spatial regions, independent continuants and dependent continuants (see the left hand side of Figure 4.2). In the remaining of this section, we focus more on the independent and dependent continuants.

Independent continuant. An *independent continuant*, in the BFO ontology, is defined as a continuant that is a bearer of quality and a realizable entity, in which other entities inhere and which itself cannot inhere in anything. Examples of independent continuants include: an organism, a person, a chair, etc. This upper-level class is extended with a set of more specific sub-clases, such as *material entity*, *object boundary*, and *site*. The *material entity* class is defined as an independent continuant that is spatially extended and whose identity is independent of that of other entities and can be maintained through time (e.g., computer).

Dependent continuant. A *dependent continuant*, in the BFO ontology, is defined as a continuant that is dependent on either one or several other independent continuant bearers or inheres in or is borne by other entities. Dependent continuants in BFO can be generically dependent or specifically dependent. A *generically dependent continuant* is defined in BFO, as a dependent continuant (A), where every instance of A requires some instance of B, but which instance of B serves can change from time to time (e.g., a PDF file that exists in different and in several hard drives). For a *specifically dependent continuant*, every instance

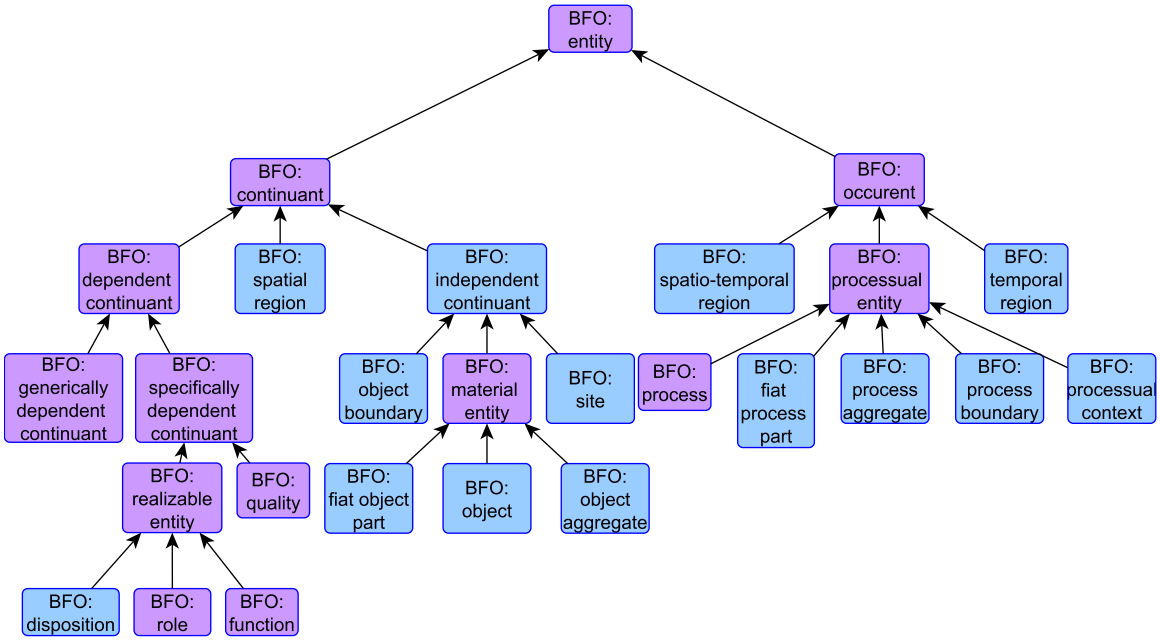


Figure 4.2: Part of the BFO ontology class hierarchy. The violet boxes represent classes that are used and further extended in the OntoDM ontology. The arrows represent is-a relations.

of A requires some specific instance of B which must always be the same (e.g., the role of being a doctor, the function of the heart in the body and others). The class *specifically dependent continuant* has two subclasses: *realizable entity* and *quality*.

Realizable entity. A *realizable entity* (RE), in the BFO ontology, is defined as a specifically dependent continuant and includes all entities that can be executed (manifested, actualized, realized) in concrete occurrences (e.g., processes). RE are entities whose instances contain periods of actualization, when they are manifested through processes in which their bearers participate. Example of realizable entity for the domain of data mining include implementations of data mining algorithms. Subclasses of RE include: *role*, *function* and *disposition*. A *role*, in the BFO ontology, is defined as a realizable entity the manifestation of which brings about some result or end that is not essential to a continuant in virtue of the kind of thing that it is, but that can be served or participated in by that kind of continuant in some kinds of natural, social or institutional contexts (e.g., training set, test set). A *function*, in the BFO ontology, is defined as a realizable entity, the manifestation of which is an essentially end-directed activity of a continuant entity in virtue of that continuant entity being a specific kind of entity in the kind or kinds of contexts that it is made for.

Quality. A *quality*, in the BFO ontology, is defined as a specifically dependent continuant that is exhibited if it inheres in an entity or any entities at all. Examples of qualities include: datatype qualities, qualities of models and patterns and others.

4.2.1.2 Occurents

In the BFO ontology, an entity that has temporal parts and that happens, unfolds or develops through time is called an *occurent* (e.g., execution of a data mining algorithm). The BFO ontology distinguishes between tree types of occurent entities: *processual entities*, *temporal regions* and *spatio-temporal regions* (see right hand side of Figure 4.2). In the context of the OntoDM ontology we extend and use only processual entities and we present them more in

detail in the remaining of this section.

Processual entity. A *processual entity* is defined as an occurrent that exists in time by occurring or happening, has temporal parts, and always involves and depends on some entity. Examples of processual entities include: the life of an organism, the course of a disease, the flight of a bird, etc. This class is further extended with the following subclasses: *process*, *fiat process part*, *process aggregate*, *process boundary*, and *processual context*. The most important processual entity class, for the ontology of data mining that we propose in this thesis, is the *process* class. In the BFO ontology, it is defined as a processual entity that is a maximally connected spatio-temporal whole and has beginnings and endings (e.g., the process of sleeping).

4.2.2 Mid-level classes: Imported ontologies

In the OntoDM ontology, we reuse some of the ontologies for representing scientific investigations and outcomes of research as mid-level ontologies (see Section 3.1) . We import and further extend classes from: the Ontology for Biomedical Investigations (OBI)[10], the Information Artifact Ontology (IAO) [11], and the Software Ontology (SWO)[12] (See Figure 4.1). The mid-level ontologies uses for the OntoDM ontology also use BFO as an upper-level ontology. Classes that are referenced and reused are imported by using the MIREOT guidelines (Courtot et al., 2011) (see Section 4.1.3). In this section, we focus more in depth on the classes imported from the IAO ontology (Section 4.2.2.1) and classes from the OBI ontology (Section 4.2.2.2).

4.2.2.1 IAO classes

In the development of an ontology of data mining, it is very important to have the ability to represent entities that deal with information, such as data, documents, models, algorithms, protocols, etc. The IAO ontology is a mid-level ontology that is dealing with information content entities (e.g., documents, file formats, specifications), processes that consume or produce information content entities (e.g., writing, documenting, measuring), material bearers of information (e.g., books, journals), and relations in which one of the relata is an information content entity (e.g., is-about, denotes, cites) (see Section 3.1.3). For this purpose, a design decision was made to incorporate and further extend some stable classes of the IAO ontology. The most important top level class in the IAO ontology is the *information content entity class* (see Fig. 4.3).

The *information content entity* (ICE), in the IAO ontology, denotes all entities that are generically dependent on some artifact and stand in relation of aboutness (IS-ABOUT) to some entity. Examples of ICE include data, narrative objects, graphs, etc. The introduction of ICE enables the representation of different ways that information relates to the world, sufficient for representing scientific investigations (and in case of OBI, specifically biomedical investigations). Subclasses of ICE include: *data item*, *textual entity*, *label*, *document*, *document part*, and *directive information entity*.

Data item and label. A *data item* is defined in the IAO ontology as an information content entity that is intended to be a truthful statement about something and is constructed/acquired by a method which reliably tends to produce truthful statements. A *label* is defined is defined in the IAO ontology as a symbol that is part of some other datum and is used to either partially define the denotation of that datum or to provide a means for identifying the datum as a member of the set of data with the same label.

[10] OBI: `http://purl.obolibrary.org/obo/obi`
[11] IAO:`http://code.google.com/p/information-artifact-ontology/`
[12] SWO: `http://theswo.sourceforge.net/`

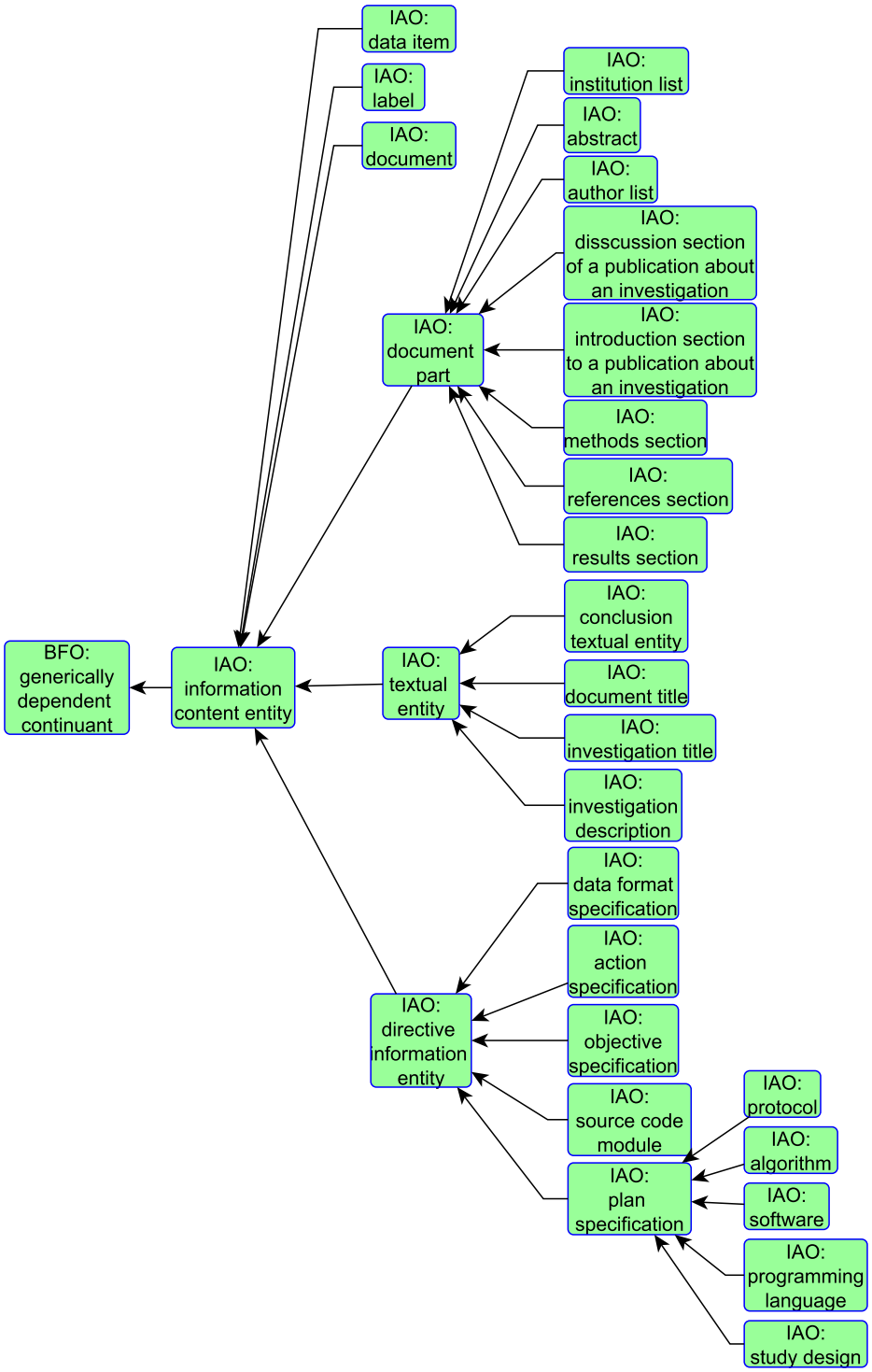


Figure 4.3: Part of the IAO class hierarchy reused and extended in the OntoDM ontology.

Document and document part. A *document* is defined in the IAO ontology as a collection of information content entities intended to be understood together as a whole, while a *document part* is defined in the IAO ontology as an information content entity that is part of a document. Document part includes the following sub-classes:

- *abstract* is defined as a summary of the entire document that is substantially smaller than the document it summarizes (it is about the document it summarizes);
- *author list* is defined as a part of a document that enumerates the authors of the document;
- *discussion section of a publication about an investigation* is defined as a part of a publication about an investigation that is about the study interpretation of the investigation;
- *institution list* is defined as a part of a document that has parts that are institution identifications associated with the authors of the document;
- *introduction section to a publication about an investigation* is defined as a part of a publication about an investigation that is about the objective specification (why the investigation is being done)
- *methods section* is defined as a part of a publication about an investigation that is about the study design of the investigation;
- *references section* is defined as a part of a document that has citations as parts; and
- *results section* is defined as a part of a publication about an investigation that is about a study design execution.

Textual entity. A *textual entity* is defined in the IAO ontology as a part of a manifestation, a generically dependent continuant whose concretizations are patterns of glyphs intended to be interpreted as words, formulas, etc. More specifically it includes the following sub-classes:

- *conclusion textual entity* is defined as a textual entity that expresses the results of reasoning about a problem, for instance as typically found towards the end of scientific papers; and
- *document title* is defined as a textual entity that names a document;

Directive informational entity. A *directive informational entity* (DIC) is an information content entity that concerns a realizable entity. DICs are ICEs whose concretizations indicate to their bearer how to realize them in a process. A directive information entity, before the OBI RC1 (release candidate 1) version, was named 'informational entity about a realizable'. Subclasses of DICs include: *data format specification*, *action specification*, *objective specification*,and *plan specification*. These are defined in the IAO ontology as follows:

- A *data format specification* is defined as the information content borne by the published document defining the specification;
- An *action specification* is defined as directive information entity that describes an action the bearer will take;
- An *objective specification* is defined as an intended process endpoint; and

- A *plan specification* is a information content entity that includes parts such as: *objective specification* and *action specifications*. When concretized, it is executed in a process in which the bearer tries to achieve the objectives, in part by taking the actions specified. A plan specification includes the following sub-classes:
  - *algorithm* is defined as plan specification which describes inputs, output of mathematical functions as well as workflow of execution for achieving an predefined objective;
  - *software* is defined as a plan specification composed of a series of instructions that can be interpreted by or directly executed by a processing unit; and
  - *programming language* is defined as a language in which source code is written, intended to executed/run by a software interpreter;

4.2.2.2 OBI classes

In the development of an ontology of data mining, it is very important to have the ability to represent some material entities, such as organism, organization, and computer. Another important representational mechanism is the ability to represent planned processes, such as documenting, planing, validation, acquisition, and others. The OBI ontology is a domain ontology for representing biomedical investigations (see Section 3.1.3). For the case of OntoDM, we use OBI as a mid-level ontology and extend it with data mining specific classes. The most important top level classes in the OBI ontology that we reuse and further extend are *material entity* and *planned process* (see Fig. 4.4).

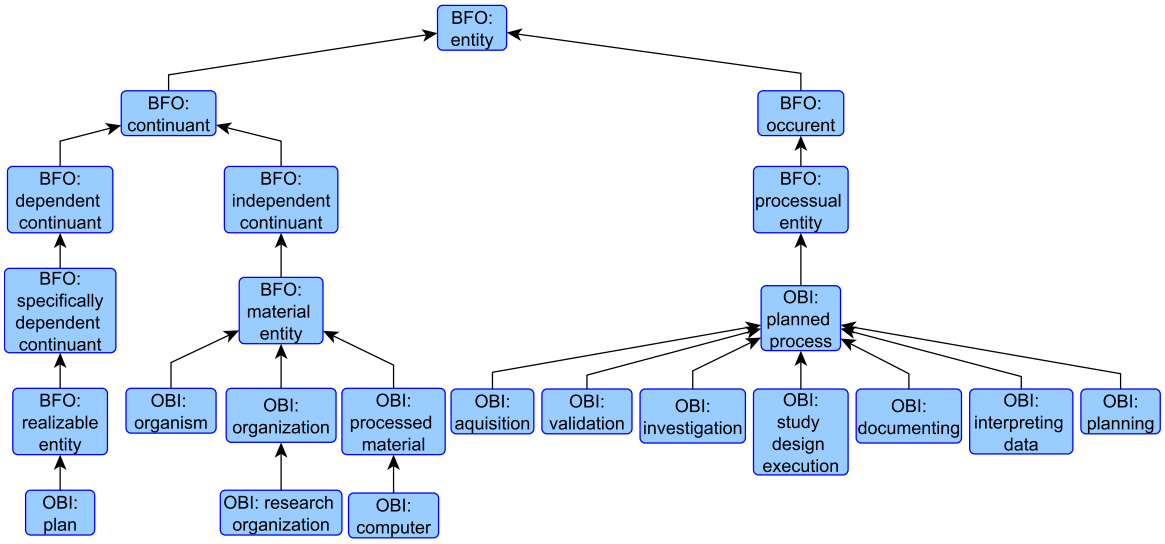


Figure 4.4: Part of the OBI class hierarchy reused and extended in the OntoDM ontology.

Material entity. In the OBI ontology, a *material entity* is an independent continuant that is spatially extended, and whose identity is independent of that of other entities and can be maintained through time. Examples of material entities include organisms, organizations, devices, etc. These are defined in the OBI ontology as follows:

- *Organism* is defined as a material entity that is an individual living system, such as animal, plant, bacteria or virus, that is capable of replicating or reproducing, growth and maintenance in the right environment;
- *Organization* is defined as a continuant entity which can play roles, has members, and has a set of organization rules. Members of organizations are either organizations themselves or individual people;

- *Processed material* is defined as a material entity that is created or changed during material processing;
- *Device* is defined as a processed material which is designed to perform some function or functions; and
- A *computer* is defined as a device which manipulates (stores, retrieves, and processes) data according to a list of instructions.

OBI Informational entities. The OBI ontology additionally includes some informational entity specific classes specifically introduced to support representation of investigations, such as protocol, study design, investigation description, investigation title, investigation identifier, and publication about an investigation. These are defined in OBI as follows:

- *protocol* is defined as a plan specification which has sufficient level of detail and quantitative information to communicate it between domain experts, so that different domain experts will reliably be able to independently reproduce the process;
- *study design* is defined as a plan specification comprised of protocols (which may specify how and what kinds of data will be gathered) that are executed as part of an investigation and is realized during a study design execution;
- *investigation description* is defined as a textual entity that describes an investigation;
- *investigation title* is defined as a textual entity that denotes an investigation;
- *investigation identifier* is defined as a symbol used to identify an investigation; and
- *publication about an investigation* is defined as a document about an investigation.

Planned process. A *planned process* is a *processual entity* that realizes a *plan*, which is the concretization of a *plan specification*, in order to achieve the objectives specified with an *objective specification*. Process entities have as participants continuants: Participants can be also active, in which that they are called agents. Examples of planned process sub-classes from OBI that are used and further extended in the case of data mining include: acquisition, validation, investigation, documenting, prediction, data transformation, data visualization, interpreting data, and planning. These are defined in the OBI ontology as follows:

- *Documenting* is a planned process of capturing information in an enduring form with the intent to communicate this information;
- *Study design execution* is a planned process that realizes the concretization of a study design;
- *Investigation* is a planned process that consists of parts such as planning, study design execution, and documentation, and which produces conclusion(s);
- *Interpreting data* is the process of evaluating the data gathered in an investigation, in the context of literature knowledge, with the objective to generate more general conclusions or to identify what additional data are necessary to draw conclusions;
- *Planning* is the process of a scientist thinking about and deciding what to use as part of a protocol for an experiment;
- *Data transformation* is a process which produces output data from input data;

- *Data visualization* is a planned process that creates images, diagrams or animations from the input data;
- *Prediction* is a process by which an event or an entity is described before it actually happens, is discovered or identified;
- *Validation* is a planned process with the objective to check that the accuracy or the quality of a claim or prediction satisfies some criteria, which is assessed by comparing with independent results; and
- *Acquisition* is the process of obtaining a continuant from a source.

4.2.3 Relations

Relations are the most essential part of a well designed ontology. It is thus crucial that the relations we use in the ontology are logically defined. At every point of ontology development, from the initial conceptualization, through the construction, to its use, all the relations introduced should not change their meaning. Consistent use of rigorous definitions of formal relations is a major step toward enabling the achievement of interoperability among ontologies and supporting automated reasoning across data derived from multiple domains.

In the design phase of constructing the OntoDM ontology, along the minimal set of fundamental built-in relations such as IS-A and INSTANCE-OF, we decided to use mainly already established and formally represented relations widely used in the biomedical domain. For this purpose, we reuse relations from the BFO 2.0 ontology, the RO ontology, and relations from other OBO Foundry ontologies, such as OBI and IAO. We additionally use several relations from the LABORS and EXACT ontologies. The complete list of relations used is presented in Table 4.2.

Table 4.2: Relations used in the OntoDM ontology.

Origin	**Relation**	**Inverse relation (if defined)**
Fundamental	IS-A	HAS-SUBCLASS
	HAS-INSTANCE	INSTANCE-OF
BFO 2.0 & OBO RO	HAS-PART	PART-OF
	HAS-PARTICIPANT	PARTICIPATES-IN
	HAS-ACTIVE-PARTICIPANT	IS-ACTIVE-PARTICIPANT-OF
	PRECEEDES	PRECEEDED-BY
	INHERES-IN	BEARER-OF
	HAS-QUALITY	IS-QUALITY-OF
	HAS-ROLE	IS-ROLE-OF
	IS-CONCRETIZED-AS	IS-CONCRETIZATION-OF
	REALIZES	IS-REALIZED-BY
IAO	IS-ABOUT	
	DENOTES	
	QUALITY-IS-SPECIFIED-AS	IS-QUALITY-SPECIFICATION-OF
OBI	ACHIEVES-PLANNED-OBJECTIVE	OBJECTIVE-ACHIEVED-BY
	HAS-SPECIFIED-INPUT	IS-SPECIFIED-INPUT-OF
	HAS-SPECIFIED-OUTPUT	IS-SPECIFIED-OUTPUT-OF
	IS-MANUFACTURED-BY	
EXACT	HAS-INFORMATION	
LABORS	HAS-REPRESENTATION	IS-REPRESENTATION-OF

In the remainder of this section, we present the set of relations used the OntoDM ontology in more detail. First, we focus on the fundamental relations. Next, we present the relations from RO and BFO 2.0. Finally, we present the relations from IAO, OBI, LABORS and EXACT.

Fundamental (built-in) relations. The fundamental relation IS-A and its inverse relation HAS-SUBCLASS are used to express the subsumption relationships between entities. The relation HAS-INSTANCE and its inverse relation IS-INSTANCE-OF connect a class with an instance of that class. The fundamental relations are formally defined by Smith et al. (2005), both at the class and at the instance level. These two relations are usually built-in relations in ontology development software packages, such as Protege.

OBO RO and BFO 2.0 relations. The OBO (Open Bio-Ontologies) Foundry design principles (see Section 4.1.2) aim to ensure interoperability between bio-ontologies (Smith et al., 2007). One of the key OBO Foundry recommendations is the use of formal relations from the OBO Relational Ontology (RO)[13] (Smith et al., 2005). Recently, the OBO RO has been significantly modified[14] and the earlier version of RO (named OBO_REL) is now deprecated[15]. The relations are now defined only at the instance level, and many new relations have been introduced, i.e., CAPABLE-OF, HAS-HABITAT, REGULATES. The OBO RO now contains only relations specific for the biomedical domains, while all domain independent relations from OBO_REL, such as HAS-PART and HAS-PARTICIPANT, are now part of BFO 2.0. Finally, BFO 2.0 now contains also some more general relations from the OBI ontology, such as REALIZES and IS-CONCRETIZED-AS.

In the OntoDM ontology, we import nine relations from BFO 2.0 and OBO RO. These are defined as follows:

- HAS-PART is defined a primitive relation between instances and a time at which the one is part of the other;
- The relations HAS-PARTICIPANT and PARTICIPATES-IN (both defined in RO) express the relationship between a process and participants in a process, that can be passive or active (in case of agents). These are primitive relations between a process, a continuant, and time;
- HAS-ACTIVE-PARTICIPANT (HAS-AGENT) is a primitive relation between a process, a continuant, and a time at which the continuant is causally active in the process;
- The relation between a dependent continuant and an entity is expressed via the relation INHERES-IN (defined in the OBI ontology and a candidate for inclusion into RO). This relation links qualities, roles, functions, dispositions and other dependent continuants to their bearers. Specializations of this relation are the following relations:
  - HAS-ROLE is a relation between a role and an entity;
  - HAS-QUALITY is a relation between a quality and an entity;
- The relation IS-CONCRETIZATION-OF (introduced by the IAO ontology) expresses the relationship between a generically dependent continuant (GDC) and a specifically dependent continuant (SCD). In the IAO ontology, this relation is defined in the following way: "A GDC may inhere in more than one entity. It does so by virtue of the fact that there is, for each entity that it inheres, a specifically dependent 'concretization' of the GDC that is specifically dependent"; and
- The relation REALIZES is used to express the relation between a process and a realizable entity, where the unfolding of the process requires execution of the realizable entity.

[13] OBO_REL: `http://purl.org/obo/owl/OBO_REL`

[14] OBO RO:`http://obo-relations.googlecode.com/svn/trunk/src/ontology/ro.owl`

[15] Differences between OBO_REL and OBO RO`http://code.google.com/p/obo-relations/wiki/DifferencesWithOldRO`

IAO relations. In the OntoDM ontology, we import several relations from the IAO ontology. These are defined as follows:

- The relation IS-ABOUT is defined to relate an information content entity to an entity and it serves as its specification;
- QUALITY-IS-SPECIFIED-AS is defined as a relation between a data item and a quality of a material entity where the material entity is the specified output of a material transformation which achieves an objective specification that indicates the intended value of the specified quality.

OBI relations. In the OntoDM ontology, we import four relations from the OBI ontology. These are defined as follows:

- The relation ACHIEVES-PLANNED-OBJECTIVE connects a planned process and a objective specification when the criteria specified in the objective specification are met at the end of the planned process;
- HAS-SPECIFIED-INPUT is a relation n between a planned process and a continuant participating in that process that is not created during the process. The presence of the continuant during the process is explicitly specified in the plan specification which the process realizes the concretization of;
- HAS-SPECIFIED-OUTPUT is a relation between a planned process and a continuant participating in that process and it is created during the process. The presence of the continuant at the end of the process is explicitly specified in the objective specification which the process realizes the concretization of; and
- IS-MANUFACTURED-BY is a relation that states that there was a process in which a continuant was built in which a person, or set of people or machines did the work.

EXACT relation. In the OntoDM ontology we reuse the relation HAS-INFORMATION defined in the EXACT ontology (Soldatova et al., 2008) to relate an agent of a process to a certain portion of information (information entity) that is essential for participating in the process.

4.2.4 Modular structure of the OntoDM ontology

For the case of the OntoDM ontology, we made a design decision to organize the ontology as a modular ontology, comprising of three ontology modules (See Figure 4.1). Each module covers a sub-domain of data mining. The three modules are as follows:

- OntoDT is an ontology module for representing knowledge about datatypes OntoDT, based on the ISO 11404 standard for general purpose datatypes (International Organization for Standardization, 2007);
- OntoDM-core is an an ontology module for representing knowledge about core data mining entities for mining structured data, based on a proposal for a general framework for data mining (Džeroski, 2007); and
- OntoDM-KDD is an ontology module for representing knowledge about data mining investigations, based on the CRISP-DM methodology (Chapman et al., 1999).

The goals, competencies, scope, implementation and description of the ontology modules are presented in detail Chapters 5, 6, and 7 respectively.

4.3 Implementation and maintenance

The OntoDM ontology and its modules OntoDT, OntoDM-core and OntoDM-KDD are implemented in OWL-DL[16] ontology language (see Section 2.4.1), a de-facto standard for representing ontologies. The ontology has been developed using the Protégé[17] ontology editor (See Section 2.4.2). We reuse the OBI consortium defined meta-data[18] to provide additional semantic annotation of the classes and relations. This includes annotations of definitions of classes, definitions sources, source ontology identification for the imported classes, and others.

By following the guidelines from other state-of-the-art ontologies (such as OBI), we provide the identifier format and deprecation policies for the OntoDM ontology. This is done in order to provide all new classes a permanent and unique ID. Additionally, we envision the necessity of keeping the list of deleted classes in the OntoDM class hierarchy under the *obsolete* class. The domain terms that have been collected so far but are still not represented in the ontology are listed under *non-curated* class in the OntoDM class hierarchy.

For the OntoDM ontology, we use one base namespace to format the URIs for the classes and relations defined first time in this ontology (`http://kt.ijs.si/panovp/OntoDM#`). The IDs of the ontology terms are diferent for each ontology module and include a combination of an ontology module ID and a five digit code:

- `OntoDT_xxxxx` for the OntoDT module
- `OntoDM_xxxxx` for the OntoDM-core module, and
- `OntoDM-KDD_xxxxx` for the OntoDM-KDD module,

The imported classes and relations in OntoDM keep the URI's from their source ontologies.

The verification of the class and relation consistency before the release is done using the Pellet[19] reasoner.

For tracking the changes in the ontology we use industry standard Subversion tool. The ontology and its modules are available online on several sources. First, the ontology can be obtained from a dedicated web page `http://www.ontodm.com`. In addition, the ontology is available from the BioPortal[20], where the BioPortal offers the possibility of obtaining ontology metrics, and additionally provides visualization capabilities.

4.4 Three-level description structure for representing entities in the data mining domain

In the previous sections, we presented the ontology best practices and general design principles, the specific design of the OntoDM ontology, and OntoDM implementation and maintenance. In this section, we describe how the introduced structure constrained by the upper-level ontology, which is used as a template, and the mid-level ontologies is used for representing entities in the data mining domain.

For the domain of data mining, we propose a three-level (horizontal) description structure. This description structure is based on the use of the upper-level ontology BFO and the extensive reuse of classes from the OBI and the IAO as mid-level ontologies. This includes three horizontal description levels: the specification level, the implementation level, and the application level. Each description level represents classes with fundamentally different nature explicitly shown with a sub-class relation to an upper-level category, and consequently

[16] OWL-DL: `http://www.w3.org/TR/owl-guide/`
[17] Protege: `http://protege.stanford.edu`
[18] OBI meta-data scheme: `http://obi-ontology.org/page/OBI_Minimal_metadata`
[19] Pellet: `http://clarkparsia.com/pellet/`
[20] BioPortal: `http://bioportal.bioontology.org/`

covers different aspects of the domain of interest. The proposed three-level (horizontal) description structure is orthogonal to the vertical ontology architecture structure which composes of an upper-level, a mid-level, and a domain level (see Figure 4.5). This means that each vertical level contains all three (horizontal) description levels.

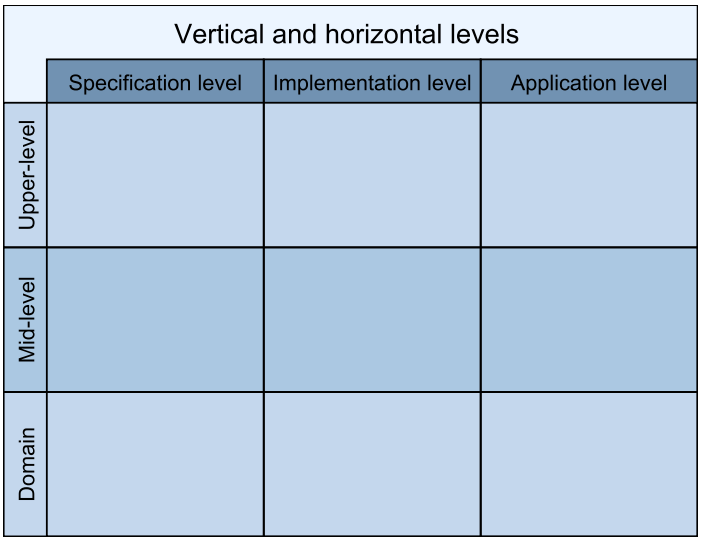
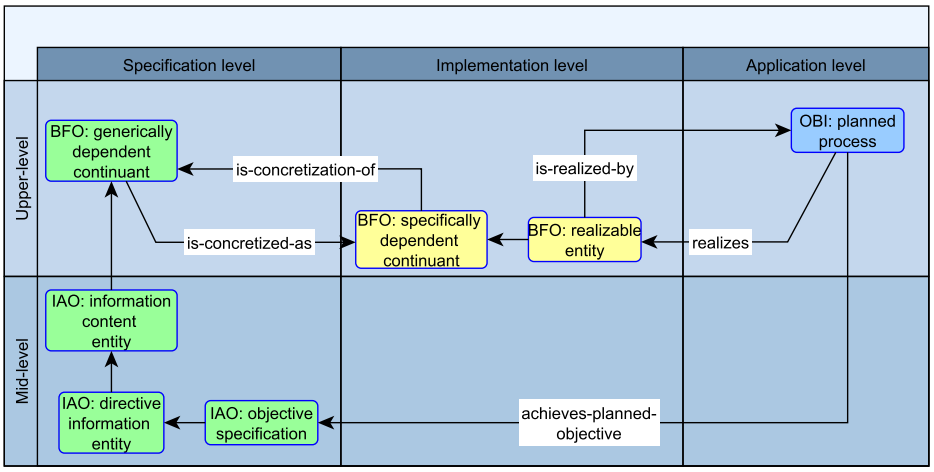


Figure 4.5: Horizontal and vertical description levels.

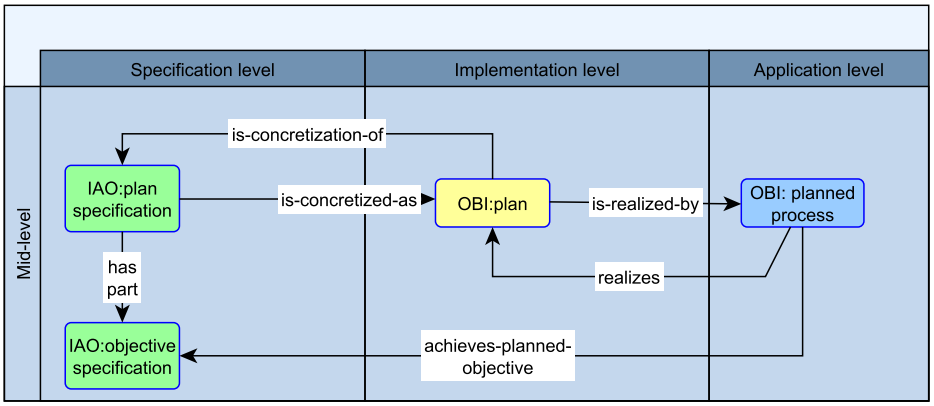
In Figure 4.6(a), we present the general case of our three horizontal description levels. The first level, named specification level, is aimed to contain and represent *generically dependent continuants* (see Section 4.2.1.1), e.g., *information content entities* (see Section 4.2.2.1). Examples of such entities in the domain of data mining are data mining tasks, algorithm specifications, dataset descriptions, generalization specifications etc. The second description level, named implementation level, is aimed to describe *specifically dependent continuants* (see Section 4.2.1.1), e.g., *realizable entities*. Examples of such entities from the domain of data mining are implementations of algorithms, implementations of components of algorithms, such as distance functions and generalizations produced by the mining process. The third level, named application level, aims at representing *planned processes*. Example of planned process entities in the domain of data mining are: the execution of a data mining algorithm and the application of a generalization on new data, among others.

The entities on each level are connected between each other using general relations, that are level independent, and level specific relations. Examples of general relations are IS-A and PART-OF: they can only be used to relate entities from the same description level. For example, an information entity (member of the specification level) can not have as parts processual entities (members of the application level). Level specific relations can be used only with entities from a specific level. For example, the relation PRECEDES is only used to relate two processual entities. The description levels are connected using cross-level relations. An entity from the specification level IS-CONCRETIZIED-AS an entity from the implementation level. Next, an implementation entity IS-REALIZED-BY an application entity. Finally, an application entity e.g., a planned process ACHIEVES-PLANNED-OBJECTIVE an objective specification, which is a specification entity.

In Figure 4.6(b), we present one of the most frequently used modeling schemas in the OntoDM ontology. A *plan specification* (see Section 4.2.2.1) is an information entity that belongs to the specification level. It has as parts *action specification* and *objective specification*. A *plan*, belonging to the implementation level, IS-A-CONCRETIZATION-OF a *plan specification*. Furthermore, a *planned process*, belonging to the application level, REALIZES a *plan*. Finally, a *planned process* ACHIEVES-PLANNED-OBJECTIVE an *objective specification*, that is a part of a plan specification.



(a) general case



(b) special case

Figure 4.6: Three-level horizontal description structure: specification, implementation and application. The rectangle objects in the figure represent ontology classes. The ontological relations are represented with directed labeled arrows. The relations that do not have an attached label are IS-A relations.

In a data mining context, a plan specification would be a data mining algorithm, that has as parts specification of actions it performs and a data mining task as an objective. A plan would be a data mining algorithm implementation, which is a concretization of a data mining algorithm. Finally, a planned process would be an execution of a data mining algorithm implementation which achieves as planned objective a data mining task.

To summarize, it is necessary to have all three aspects represented separately in the ontology as they have distinctly different nature. This will facilitate different uses of the ontology. For example, in the data mining context, the specification aspect can be used to reason about data mining algorithms and its components; the implementation aspect can be used for search over implementations of data mining algorithms and to compare various implementations; and the application aspect can be used for constructing data mining scenarios and workflows, definition of participants of a workflow and its parts.

5 OntoDT: Ontology Module for Datatypes

In this chapter, we present the OntoDT ontology module for describing and representing general-purpose datatypes. First, we present the OntoDT goal, scope and competencies (Section 5.1). Next, we discuss the implementation of the OntoDT-core ontology module (Section 5.2). Finally, we describe in more detail the structure of this ontology module (Section 5.3).

5.1 Goal, scope and competencies

In the context of the development of the ontology of data mining that needs to be general enough to allow the representation of mining structured data, we developed a separate ontology module, named OntoDT, for representing the knowledge about datatypes. In the preliminary OntoDT development phase, the classes used to represent datatypes were integrated in OntoDM-core (Panov et al., 2010). For generality and reuse purposes, however, we later exported datatype specific OntoDM classes in a separate ontology module - OntoDT. This ontology module can now be reused independently by any other ontology that requires a representation of and reasoning about general purpose datatypes.

The content of the OntoDT ontology module is based on an ISO standard for the representation of datatypes in computer systems and programming languages. The first edition of the standard, named 'Language independent datatypes', was published in 1996 (International Organization for Standardization, 1996). The revised version, named 'General-Purpose Datatypes', was published in 2007 (International Organization for Standardization, 2007). The standard specifies both primitive datatypes, defined without a reference to other datatypes, and non-primitive datatypes, defined in terms of other datatypes, that occur commonly in programming languages. The definitions of datatypes are independent of any particular programming language or implementation. Furthermore, Meek (1994) discusses a proposal for a taxonomy of datatypes using as a base the first version of the ISO standard (International Organization for Standardization, 1996). His taxonomy follows the practice of starting with a number of primitive datatypes and using these to construct others. The proposed taxonomy is given only in the form of an overview and a discussion of how things are done, without any formal representation (in a machine processable language) that can be reused further.

Based on the main aim of OntoDT and the OntoDM ontology desiderata, we established a list of competency questions our ontology module is designed to answer. The most important competency questions are listed in Table 5.1. Judging from the list of questions, the ontology will include information on datatypes, datatype properties, characterizing operations on datatypes, and different types of datatypes, such as primitive datatypes and generated datatypes.

Table 5.1: OntoDT competency questions.

Q_n	**Query**
1	What is the set of characterizing operations for a datatype X?
2	What is the set of datatype qualities for a datatype X?
3	What is the value space for a datatype X?
4	What is the set of datatypes that have a datatype quality X?
5	What is the set of datatypes that have a characterizing operation X?
6	What is the set of datatypes that have a datatype quality X and characterizing operation Y?
7	What are the aggregated datatypes that have an aggregate generator property X?
8	What is the set of aggregate properties for an aggregate datatype X?
9	What are the field components for a tuple datatype X?
10	What is the base datatype for a set/bag/sequence datatype X?
11	What is the base datatype for a subtype X?
12	What is the subtype generator for a subtype X?
13	What is the set of subtypes that have datatype X as their base datatype?
14	What is the set of subtypes that are generated by a subtype generator X?

5.2 Implementation

In order to represent datatypes, we model the ISO standard (International Organization for Standardization, 2007) within the OntoDT ontology module by reusing and extending upper level classes from the IAO ontology. This is compliant with the design principles discussed in Chapter 4. The OntoDT ontology module contains entities from the specification and implementation description level. At this phase of development, the ontology module does not contain entities from the application level. The specification level (see Section 5.2.1) contains information entities needed to describe and represent the datatypes. The implementation level (see Section 5.2.2) contains quality entities needed to characterize the datatypes.

5.2.1 Specification level

The specification level of OntoDT (see Figure 5.1) consists of classes that are extensions of the IAO class *information content entity*. At the top level, the specification level includes classes for representing data such as *specification entity* and *directive information entity*. These top level classes are further sub-classed in order to provide representational mechanism to support a representation of mining structured data.

For example, in OntoDT we extend the IAO *directive information entity* class with *data representational model* from the OBI ontology in order to provide a mechanism for repre-

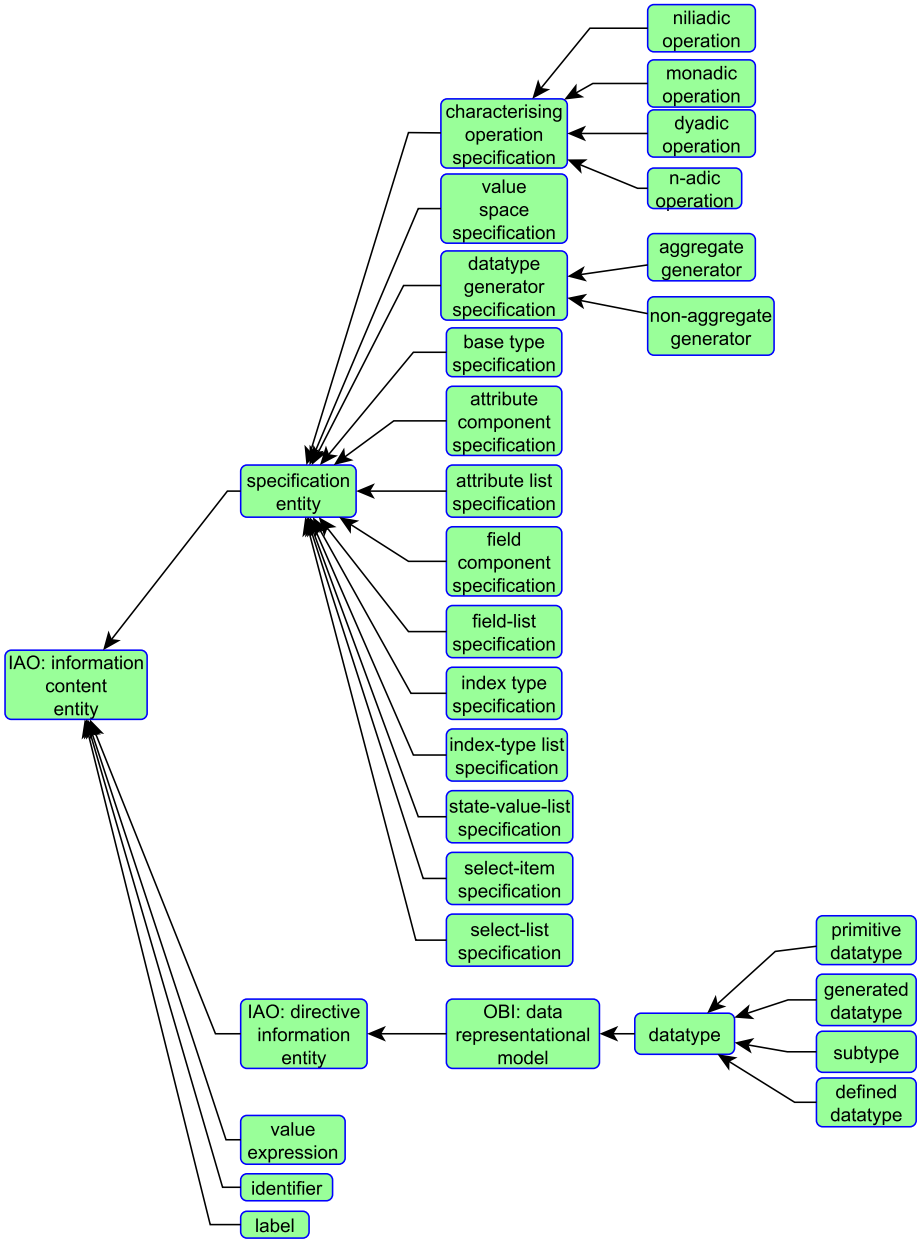


Figure 5.1: Part of the class taxonomy of the OntoDT specification level.

senting datatypes. Furthermore, we introduce a *specification entity* class as an information entity that represents an explicit description of requirements to be satisfied by a material, product, or service. In the context of datatypes, we further define sub-classes of specification entities that are used to characterise the general intrinsic properties of datatypes, such as *value space specification*, *characterizing operation specification*. In addition, represent supporting entities for describing specific types of datatypes, such as *field component specification* and *field-list specification*, used for describing a tuple datatype.

5.2.2 Implementation level

The implementation level of OntoDT (see Figure 5.2) consists of classes that are extensions of the BFO class *quality*. We extend the *quality* class with *information content quality*, as a specific class that covers qualities of information entities. In the context of datatypes, this class is further extended with qualities of datatypes (*datatype quality* class) and qualities of aggregate generators (*aggregate generator quality* class).

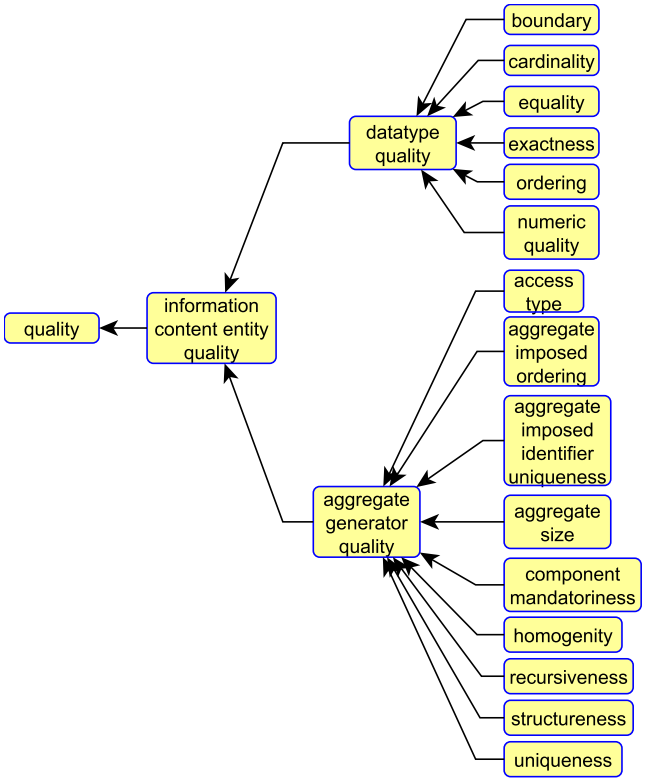


Figure 5.2: Part of the class taxonomy of the OntoDT implementation level.

5.3 Ontology module description

In this section, we present an overview of the key classes for representing knowledge about datatypes, having in mind the two different description levels and the competency questions the ontology module is designed to answer (see Table 5.1). First, we present the datatype class, and classes for characterizing datatypes such as value space, datatype qualities, and characterizing operations (Section 5.3.1). Next, we present the OntoDT taxonomy of datatypes, which includes primitive datatypes (Section 5.3.2), generated datatypes (Section 5.3.3), subtypes (Section 5.3.4), and defined datatypes (Section 5.3.5).

5.3.1 Datatype

A *datatype* IS-A *data representational model* that denotes a type of data, with the set of distinct values that it can take, the properties of those values, and the operations on those values. The *datatype* class (see Figure 5.3) has as parts a *value space specification* and a set of *characterizing operations specifications* . Finally, a datatype has as qualities *datatype qualities*.

Value space specification. A *value space specification* IS-A *specification entity* that specifies the collection of values for a given datatype. The value space of a given datatype can be defined in several different ways: by enumerating the values; with axioms using a set of fundamental notions; as a subset of values defined in another value space with a given set of properties; or as a combination of arbitrary values from some other defined value space by specifying the construction procedure.

A value space of a datatype can contain regular or sentinel values. Regular values are consistent with the datatype properties and characterizing operations, while sentinel values are elements that belong to the datatype, but are not completely consistent with

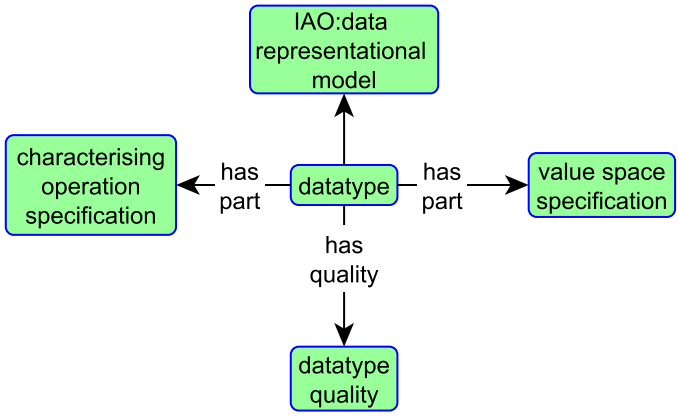


Figure 5.3: The datatype class in OntoDT.

the datatype properties and characterizing operations. Examples of sentinel values include: not-a-number, not-applicable, infinity, etc.

Characterizing operations. A *characterizing operation specification* IS-A *specification entity* that specifies the operations on the datatype which distinguish a datatype from other datatypes with identical value spaces. The OntoDT ontology module contains operation specification instances for all defined primitive and generated datatypes. The characterizing operation specification on a datatype can be: niliadic, monadic, dyadic and n-adic (see Figure 5.4), depending on the number of input arguments for the operations.

A *niliadic operation* specifies an operation that yields values of a given datatype. Examples of niliadic operations instances include operations for testing if a datatype contains any members, such as `Empty:bag`, `Empty:set`, and `Empty:sequence`. A *monadic operation* specifies an operation that maps a value of a given datatype into another value of the given datatype, or into a value of the *boolean* datatype. Examples of monadic operations include `Not:boolean`, `Select:set`, `Successor:enumerated`, `FieldSelect:tuple`, and others. A *dyadic operation* specifies an operation that maps a pair of values of a given datatype into another value of the given datatype or into a value of the *boolean* datatype. Examples of dyadic operations include `Equal:tuple`, `Add:real`, `InOrder:enumerated`, `And:boolean`, and others. A *n-adic operation* specifies an operation that maps an ordered n-tuple of values, each of which is of a specific datatype, into values of another given datatype. Examples of n-adic operations include the `Replace:array` operation.

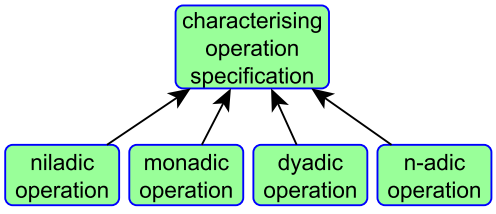


Figure 5.4: Characterizing operations in OntoDT.

Datatype qualities A *datatype quality* IS-A *information content quality* that specifies the intrinsic properties of the data units represented by the datatype, regardless of the properties of their representations in computer systems. Each datatype has as part a set of datatype qualities. These include qualities such as: *equality*, *order*, *bound*, *cardinality*, *exactness*, and *numeric property* (See Figure 5.5). All classes have defined instances. For example, the

Table 5.2: List of datatype qualities.

Quality	Description
equality	The notion of equality is valid for every value space, by fulfilling a set of predefined criteria.
ordering	A datatype is ordered, if there exists an order relation defined on its value space.
boundary	A datatype is bounded above, if it is ordered and there is an upper bound value U, such that all values s in the value space $s \leq U$. A datatype is bounded below, if it is ordered and there is a lower bound value L, such that all values s in the value space $L \leq s$. A datatype is bounded, if the value space has both an upper and lower bound.
cardinality	A value space has the notion of cardinality. The cardinality can be: finite, countable and uncountable. A datatype has the cardinality of its value space.
exactness	A datatype is exact, if every value in the value space is distinguishable from every other value in the value space, then the datatype is exact. If the datatype is not exact, then it is approximate.
numeric quality	A datatype is numeric if its values are quantities expressed in some mathematical number system.

boundary quality class has the following instances: `bounded`, `unbounded`, `bounded below`, `bounded above`, `unbounded below`, and `unbounded above`. The list of datatype qualities with a short description is presented in Table 5.2.

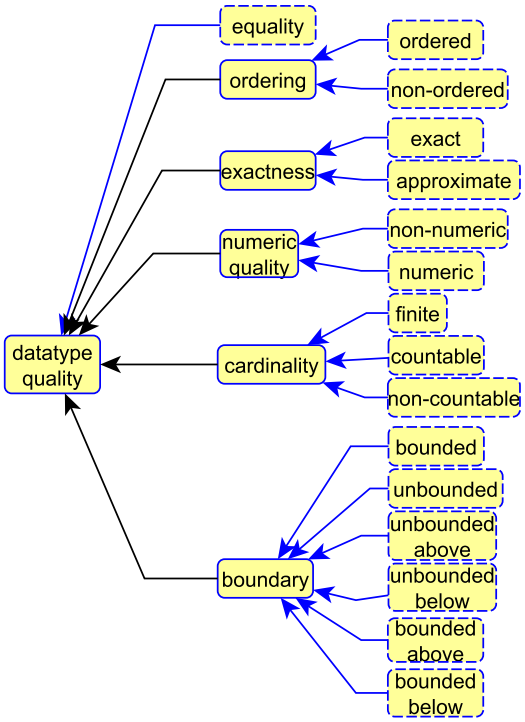


Figure 5.5: Datatype quality classes and instances in OntoDT. The black arrows denote IS-A relations, while the blue arrows denote IS-INSTANCE-OF relation.

5.3.1.1 The OntoDT datatype taxonomy

In the OntoDT ontology, in line with the ISO/IEC 11404 standard for datatypes, we define the major classes of datatypes using four different formal methods (See Figure 5.6). First, we define the *primitive datatypes* by explicit specification. The primitive datatypes have well defined abstract notions and are independent of other datatypes. Next, we define the *generated datatypes* by implicit specification. The generated datatypes are syntactically

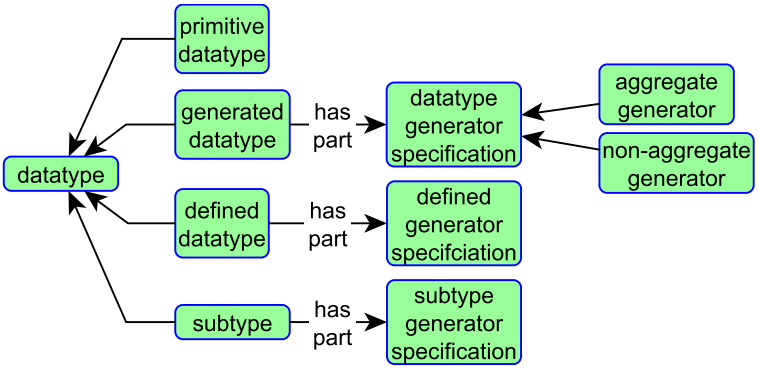


Figure 5.6: The OntoDT datatype taxonomy.

and semantically dependent on other datatypes, and are specified implicitly with *datatype generators*, which embody independent abstract notions. Furthermore, we define the *defined datatypes* by a specification with a datatype declaration. The defined datatypes allow defining additional identifiers and refinements to both primitive and generated datatypes. Finally, for all datatypes defined by any of the foregoing ways, we can define a *subtype* by providing an explicit subtype specification.

5.3.2 Primitive datatype

A *primitive datatype* IS-A *datatype* whose value space is defined either axiomatically or by enumeration. All primitive datatypes are conceptually atomic and therefore are defined in terms of well defined abstract notions. The OntoDT ontology defines both instances and classes of primitive datatypes.

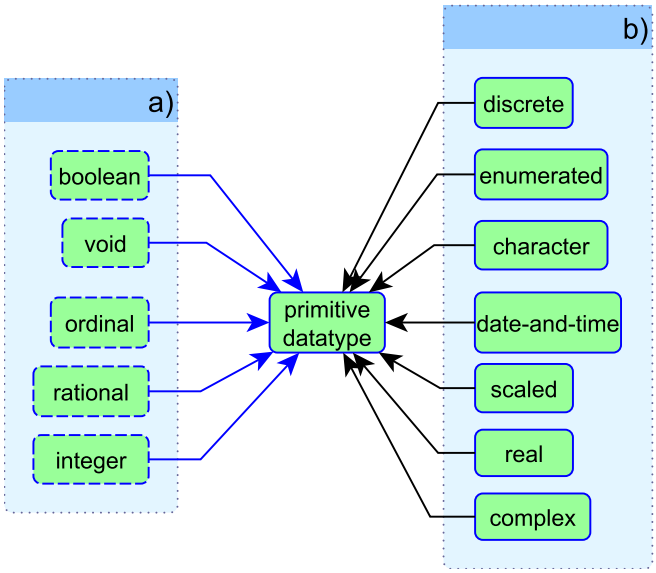


Figure 5.7: Instances and subclasses of primitive datatypes.

Instances of primitive datatypes. Instances of primitive datatypes in OntoDT include: `boolean`, `integer`, `ordinal`, `rational` and `void` (See Figure 5.7a). These are defined as follows:

- `Boolean` is the mathematical datatype associated with two-valued logic;

- `Integer` is the mathematical datatype composed of the exact integer values;
- `Ordinal` is the datatype of the ordinal numbers, as distinct from integer, it is the infinite enumerated datatype;
- `Rational` is the mathematical datatype comprising the rational numbers; and
- `Void` is the datatype specifying an object whose presence is required, but carries no information.

Table 5.3: Operations for predefined instances of primitive datatypes.

`boolean`	`ordinal`	`integer`	`rational`	`void`
Equal	Equal	Equal	Equal	Equal
Not	InOrder	InOrder	NonNegative	
And	Successor	NonNegative	InOrder	
Or		Negate	Promote	
		Add	Add	
		Multiply	Negate	
			Multiply	
			Reciprocal	

According to the definition of a datatype from Section 5.3.1, each instance of a primitive datatype has a set of datatype properties and a set of characterizing operations. The set of characterizing operations for the instances is presented in Table 5.3 and the set of datatype properties in Table 5.4.

Table 5.4: Properties of primitive datatype instances.

Property		`Boolean.`	`Ordinal`	`Integer`	`Rational`	`Void`
equality		✓	✓	✓	✓	✓
order	ordered		✓	✓	✓	
	unordered	✓				
bound	b. above					
	b.below		✓			
	bounded					
	u. below					
	u. above		✓			
	unbounded			✓	✓	
exactness	exact	✓	✓	✓	✓	
	approximate					
numeric	numeric			✓	✓	
	non-numeric	✓	✓			

For example, the *ordinal* datatype (See Figure 5.8) has the *equality* property; it is *bounded below* and *unbounded above*; it is *exact*; and it is *non-numeric*. Furthermore, the ordinal datatype has the following set of characterizing operations: *Equal*, *InOrder* and *Successor*.

Classes of primitive datatypes. The sub-classes of the primitive datatypes include: *discrete*, *enumerated*, *character*, *date-and-time*, *scaled*, *real*, and *complex*. These are defined as follows:

- *Discrete* (or State as defined in the ISO standard) defines a class of datatypes, whose instances comprise a finite number of distinguished but unordered values;
- *Enumerated* defines a class of datatypes, whose instances comprise a finite number of distinguished values having an intrinsic order;

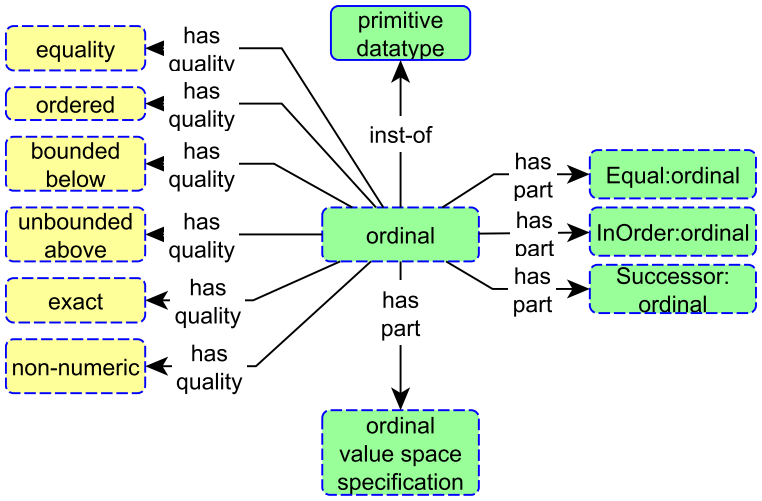


Figure 5.8: The ordinal datatype class in OntoDT.

- *Character* defines a class of datatypes, whose value spaces are character sets;
- *Date-and-time* defines a class of datatypes, whose values are points in time to various common resolutions;
- *Scaled* defines a class of datatypes, whose value spaces are subsets of the rational value space, each individual instance having a fixed denominator: Datatypes from this class posses the concept of approximate value;
- *Real* defines a class of datatypes, whose instances are computational approximations of the mathematical datatype comprising the real numbers; and
- *Complex* defines a class of datatypes, whose instances are computational approximations to the mathematical datatype comprising the complex numbers.

The classes of primitive datatypes can be further instantiated by specifying additional parameters that are different for each class of primitive datatypes.

Table 5.5: Operations for sub-classes of primitive datatype.

state	**enumerated**	**character**	**date&time**	**scaled**	**real**	**complex**
Equal	Equal	Equal	Equal	Equal	Equal	Equal
	InOrder		InOrder	InOrder	InOrder	Promote
	Successor		Difference	Negate	Promote	Negate
			Round	Add	Negate	Add
			Extend	Round	Add	Multiply
				Multiply	Multiply	Reciprocal
				Divide	Reciprocal	SquareRoot

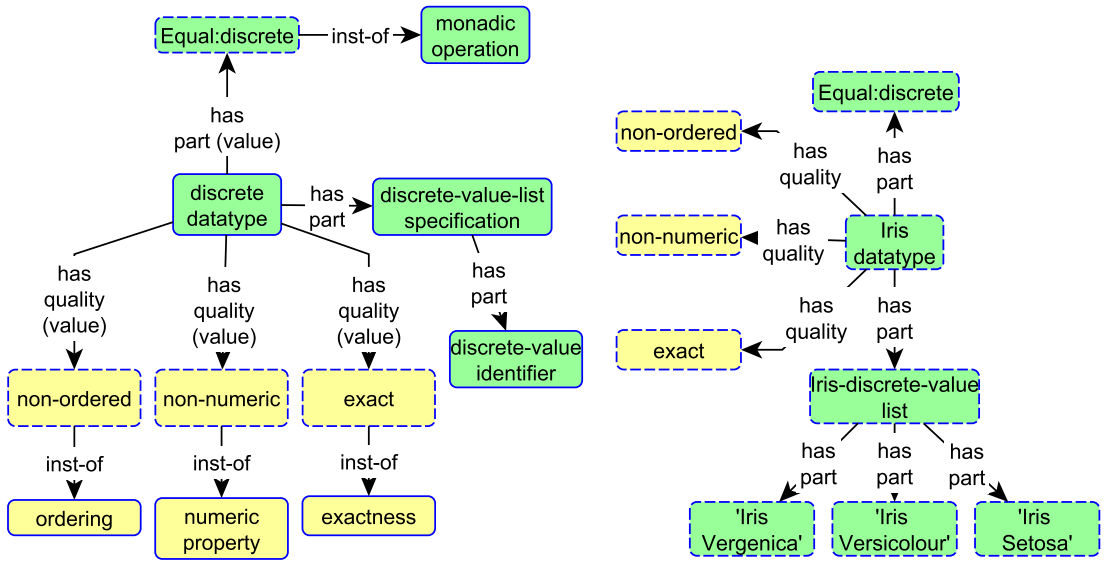
According to the definition of a datatype from Section 5.3.1, each primitive datatype class has a set of datatype properties and a set of characterizing operations. The set of characterizing operations for the subclasses of primitive datatypes is presented in Table 5.5 and the set of datatype properties in Table 5.6.

In Figure 5.9(a), we present an example representation of a discrete datatype. The datatype instances of the class discrete have the following datatype properties: `unordered`, `non-numeric` and `exact`. All instances of a discrete datatype have `Equal` as a characterizing operation. They differ in the *discrete-value-list specification*. The *discrete-value-list specification* IS-A *specification entity* that specifies the *discrete-value identifiers* for the datatype.

Table 5.6: Properties of classes of primitive datatypes.

Property		State	Enum.	Char.	Date.	Scaled	Real	Compl.
equality		✓	✓	✓	✓	✓	✓	✓
order	ordered		✓		✓	✓	✓	
	unordered	✓						✓
bound	b. above							
	b.below							
	bounded		✓	✓				
	u. below							
	u. above							
	unbounded				✓	✓	✓	
exactness	exact	✓	✓	✓	✓	✓		
	approximate						✓	✓
numeric	numeric					✓	✓	✓
	non-numeric	✓	✓	✓	✓			

For example, in Figure 5.9(b), we present the representation of the `Iris datatype` instance. The `Iris datatype` has a discrete-value-list specification that includes three state identifiers: `'Iris Setosa'`, `'Iris Versicolor'`, and `'Iris Virginica'`.



(a) class view (b) instance view

Figure 5.9: Example representation of a discrete datatype in OntoDT.

5.3.3 Generated datatype

A *generated datatype* IS-A *datatype* that is defined with a *datatype generator specification*. A *datatype generator specification* IS-A *specification entity* that specifies the conceptual operation on one or more datatypes which yields a datatype. This entity specifies the criteria for the number and properties of datatypes to be operated upon. Next, it defines a construction procedure which creates a new value space from the value space of the element datatypes. Furthermore, it specifies the set of characterizing operations for the resulting datatype. Finally, the collection of datatypes to which the datatype generator was applied to are called its parametric datatypes: A datatype generator can be applied to many different parametric datatypes.

In general, we distinguish two classes of generators: *non-aggregate generators* and *aggregate generators* (see Figure 5.10). Non-aggregate generators generate datatypes whose

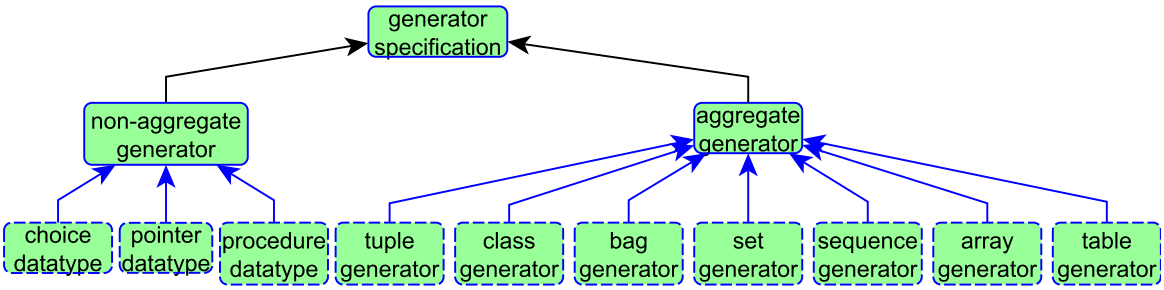


Figure 5.10: The class generator specification in OntoDT.

values are atomic. Examples of non-aggregate generator instances include: `choice generator`, `pointer generator`, and `procedure generator`. Aggregate generators generate datatypes whose values can be decomposed. Examples of aggregate generator instances include: `tuple generator`, `bag generator`, `set generator`, `sequence generator`, `array generator`, and `table generator`. Finally, having two classes of generators leads to a similar classification of generated datatypes (See Figure 5.11): non-aggregate datatypes and aggregate datatypes.

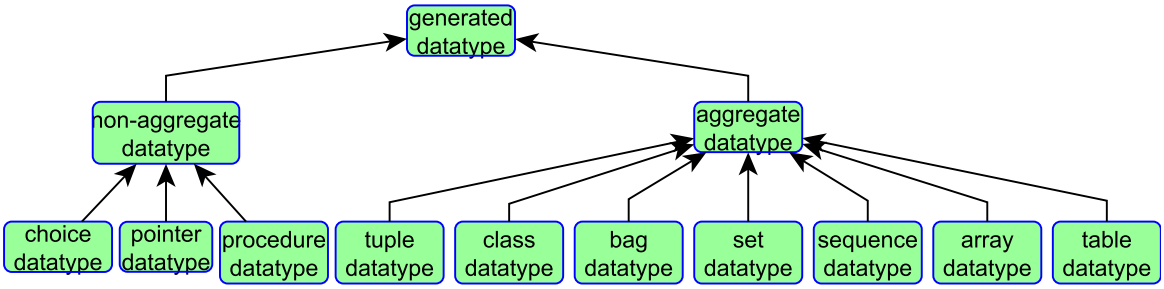


Figure 5.11: The taxonomy of generated datatypes in OntoDT.

Non-aggregate datatype. A *non-aggregate datatype* IS-A *generated datatype* that is specified by a *non-aggregate generator*. Each generator specifies the procedure of generating a class of datatypes (see Fig. 5.11). For example:

- A *choice generator* specifies the procedure for generating the *choice* datatype, each of whose values is a single value from any of a set of alternative datatypes;
- A *pointer generator* specifies the procedure for generating the *pointer* datatype, each of whose values constitutes a means of reference to values of another datatype and are atomic; and
- A *procedure generator* specifies the procedure for generating the *procedure* datatype, each of whose values is an operation on values of other datatypes and it is atomic.

Aggregate datatype. An *aggregate datatype* (or a structured datatype) IS-A *generated datatype*, each of whose values is made up of values of other datatypes, called parametric or component datatypes, joined together by an *aggregate generator* (See Figure 5.12). An *aggregate generator* IS-A *datatype generator specification* that specifies the algorithmic procedure applied to the value spaces of the component datatypes to yield the value space of the aggregate datatype, as well as a set of characterizing operations specific to the generator. Each aggregate generator instance defines a separate class of aggregate datatype (see Figure 5.11), such as: *tuple datatype*, *class datatype*, *bag datatype*, *set datatype*, *sequence*

datatype, *array datatype*, and *table datatype*. The component values of an aggregate value are accessible through characterizing operations. The sets of characterizing operations for aggregate datatypes are given in Table 5.7.

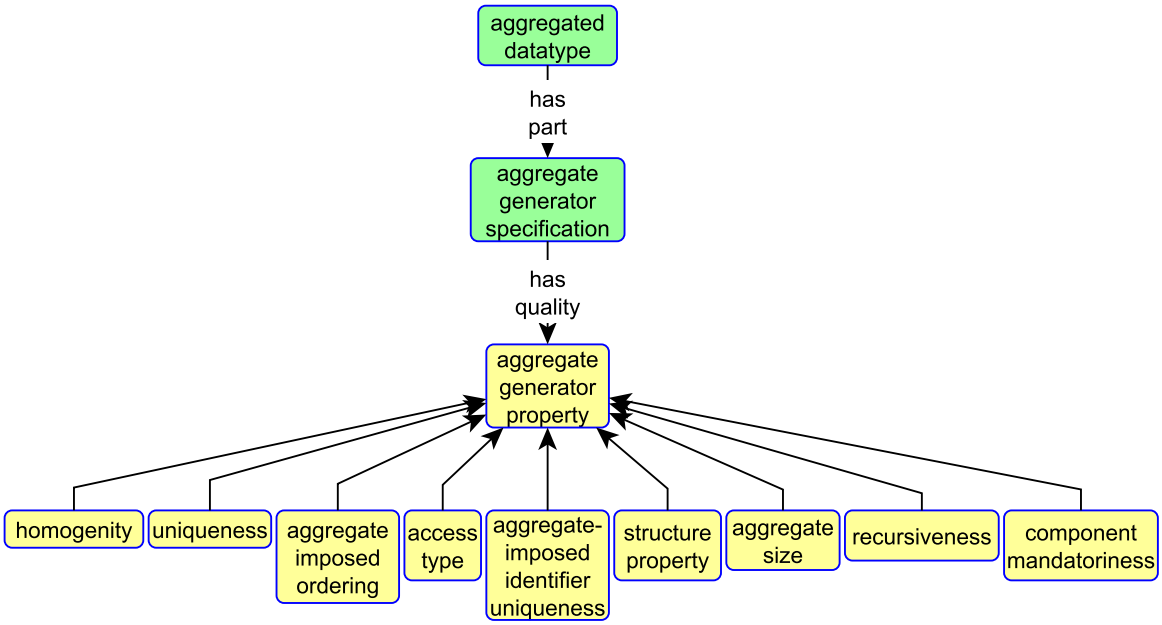


Figure 5.12: The aggregated datatype class in OntoDT.

Table 5.7: Characterizing operation instances for the aggregate datatypes.

tuple	**class**	**set**	**bag**	**sequence**	**array**	**table**	
Equal	Equal	IsIn	IsEmpty	isEmpty	Select	MapToBag	
FieldSel.	AttributeSel.	Subset	Equal	Head	Equal	MaptoTable	
FieldRepl.	AttributeRepl.	Equal	Empty	Tail	Replace	Serialize	
		Difference	Serialize	Equal		IsEmpty	
		Union	Select	Empty		Equal	
		Intersection	Delete	Append		Empty	
		Empty	Insert			Delete	
		SetOf				Insert	
		Select				Select	
						Fetch	

Aggregate datatypes are distinguished from one another by qualities that describe: the relationships among the component datatypes; the relation between each component and the aggregate; and the sets of characterizing operations. The qualities specific to an aggregate are independent of the qualities of the component datatypes. These include qualities such as: *homogeneity*, *size*, *uniqueness*, *aggregate-imposed identifier uniqueness*, *aggregate-imposed ordering*, *access method*, *recursiveness*, *structured property*, and *mandatory component property* (See Figure 5.12). The quality of aggregate generators for each generator instance is given in Table 5.8. In the remainder of this section, we give a brief overview of the major aggregate datatype classes.

Tuple datatype. The *tuple generator* (also named record generator in the ISO standard) specifies the procedure for generating a *tuple datatype*. The values of the tuple datatype are heterogeneous aggregations of values of component datatypes. Each aggregation has one value for each component datatype. The component datatypes are keyed by an identifier.

Figure 5.13 gives the representation of the tuple datatype in OntoDT. A tuple datatype (see Figure 5.13(a)) is an aggregate datatype that has as parts a tuple generator, a field-list

Table 5.8: Aggregate generator qualities.

Property		**Tuple**	**Class**	**Set**	**Bag**	**Sequence**	**Array**	**Table**
homogeneity	homogeneous			✓	✓	✓	✓	
	heterogeneous	✓	✓					✓
size	fixed	✓	✓				✓	
	variable			✓	✓	✓		✓
uniqueness				✓				
size	fixed	✓	✓				✓	
	variable			✓	✓	✓		✓
aggregate-imposed identifier uniqueness				✓				
aggregate-imposed ordering						✓		
access method	direct	key	key				index	key
	indirect			value	✓	position		

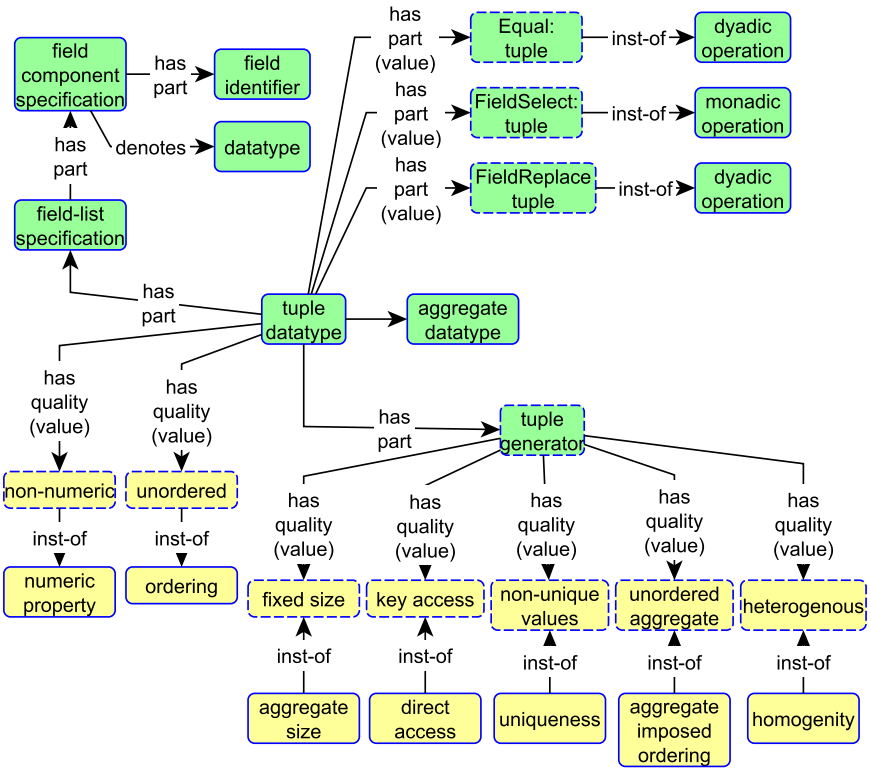
specification and a set of characterizing operations (e.g., `Equal`, `FieldSelect` and `Field Replace`). In addition, it has as qualities a set of datatype qualities (e.g., `non-numeric`, `unordered`). The tuple generator has a set of aggregate generator properties. These include instances such as: `fixed size`, `key access`, `non-unique values`, `unordered aggregate`, and `heterogeneous`. The field-list specification is a specification entity that specifies the list of field components. Each field component specification contains a unique identifier of the component, and its datatype.

In Figure 5.13(b), we present an instance of the tuple datatype. `Iris-tuple` is an instance of the tuple datatype. The `Iris-tuple` instance inherits all the characterizing operations, the datatype properties and the record generator from the parent class. In addition, the `Iris-tuple` datatype has a specification of the component datatypes. For example, the `Iris field-list specification` has as parts `Iris field-list component` instances for each component datatype. Each component specification includes an identifier (e.g., `'sepal length'`) and denotes the datatype of the component (e.g., `real(f:def,r:def)`).

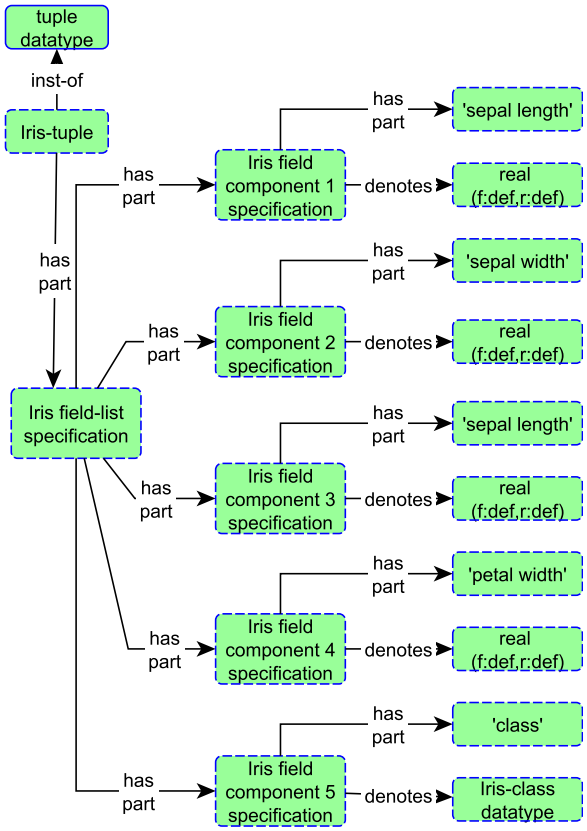
Class datatype. In OntoDT, the *class generator* specifies the procedure for generating a *class datatype*. The values of the class datatype are heterogeneous aggregations of values of the component datatypes. Unlike the tuple datatype, the components of a class datatype may include procedure datatypes. Each aggregation has one value for each component datatype. The component datatypes are keyed by an identifier.

Set datatype. In OntoDT, the *set generator* specifies the procedure for generating a *set datatype*. The set datatype is defined with a *base specification*, that specifies the component datatype (in this case named element datatype). The value-space of the set datatype is the set of all subsets of the value-space of the element datatype. The operations defined on the set datatype are equal to the operations appropriate to the definition of a mathematical set.

In Figure 5.14, we present the representation of the set datatype in OntoDT. A set datatype (see Figure 5.14(a)) is an aggregate datatype that has as parts a set generator, a base specification and a set of characterizing operations (e.g., `Difference`, `Empty`, `Equal`, `Intersection`, `IsIn`, `Select`, `SetOf`, `Subsect`, and `Union`). Additionally, it has as qualities a set of datatype qualities (e.g., `exact`, `non-numeric`, `unordered`).

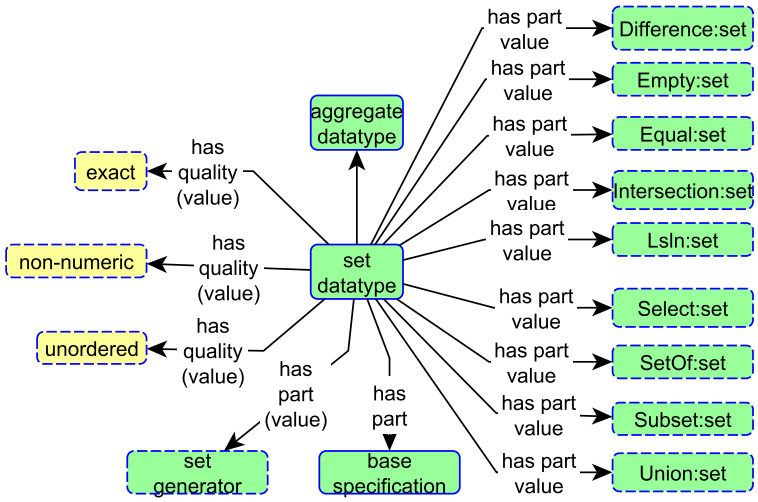


(a) class view

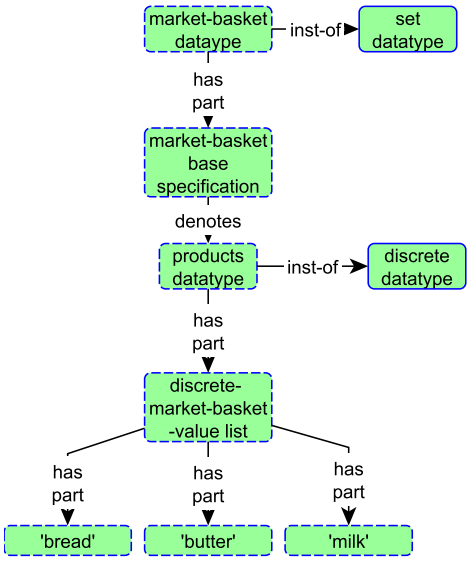


(b) instance view

Figure 5.13: The tuple datatype class in OntoDT.



(a) class view



(b) instance view

Figure 5.14: The set datatype and an example instance thereof in OntoDT.

In Figure 5.14(b), we present an instance of the set datatype. `Market-basket datatype` is an instance of the set datatype. The market-basket datatype instance inherits all the characterizing operations, the datatype properties and the set generator from the parent class. In addition, the `market-basket datatype` has as `market-basket base specification`. It denotes the datatype of the components of the set. In this case, it denotes the `products datatype`. It is an instance of the discrete datatype and has a value list containing the values `'bread'`, `'butter'`, and `'milk'`.

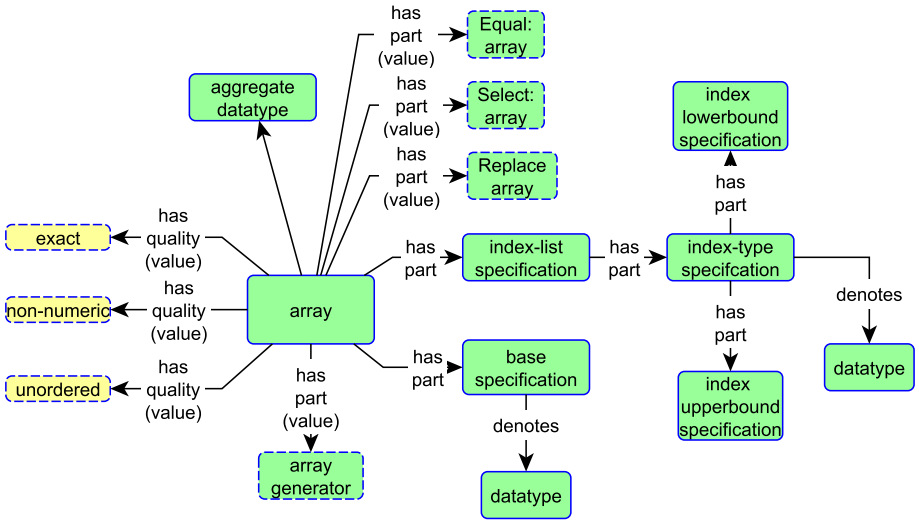
Bag datatype. In OntoDT, the *bag generator* specifies the procedure for generating a *bag datatype*. The bag datatype is defined with a *base specification* that specifies the element datatype, the same as in the set datatype. The values of the bag datatype are collections of instances of values from the element datatype. Unlike in the set datatype, where a value can appear only once, multiple instances of the same value may occur in a given bag.

Sequence datatype. In OntoDT, the *sequence generator* specifies the procedure for generating a *sequence datatype*. The sequence datatype is defined with a *base specification*, that

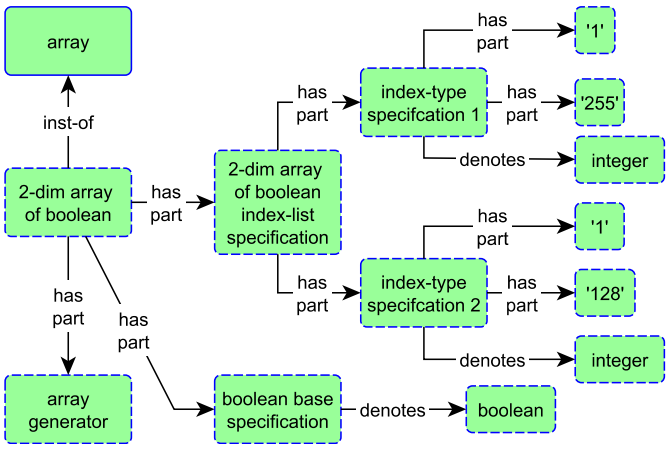
specifies the element datatype. The values of the sequence datatype are ordered sequences of values from the element datatype. The ordering is imposed on the values, and not intrinsic in the underlying datatype. In a sequence, the same value may occur more than once.

Array datatype. In OntoDT, the *array generator* specifies the procedure for generating an *array datatype*. The values of the array datatype are associations between the product space of one or more finite datatypes, designated the index datatypes (specified by the *index-type list specification*), and the value space of the element datatype (specified by the *base specification*). Every value in the product space of the index datatypes associates to exactly one value of the element datatype.

Figure 5.15, gives the representation of the array datatype in OntoDT. An array datatype (see Figure 5.15(a)) is an aggregate datatype that has as parts an *array generator*, an *index-type list specification*, a *base specification*, and a set of characterizing operations (e.g., `Equal`, `Select`, `Replace`). In addition, it has as qualities a set of datatype properties (e.g., `exact`, `non-numeric`, `unordered`). The *index-type list specification* is used to specify the index datatype, and the upper/lower bounds of the index.



(a) class view



(b) instance view

Figure 5.15: The representation of the array datatype class and an instance thereof in OntoDT.

In Figure 5.15(b), we show the representation of an instance of the array datatype. `2-dim array of boolean` is an instance of the array datatype. The 2-dim array of boolean datatype instance inherits all the characterizing operations, the datatype properties and the array generator from the parent class. In addition, the `2-dim array of bolean` datatype has as `2-dim array of boolean index-list specification`. It denotes the index datatypes. For example, the `index-type specification 1` denotes `integer` as an index datatype with 1 as a lower and 255 as an upper index bound. The boolean base specification denotes `boolean` as the base datatype.

Table datatype. In OntoDT, the *table generator* specifies the procedure for generating a *table datatype*. The values of the table datatype are collections of values in the product space of one or more field datatypes, such that each value in the product space represents an association among the values of its fields.

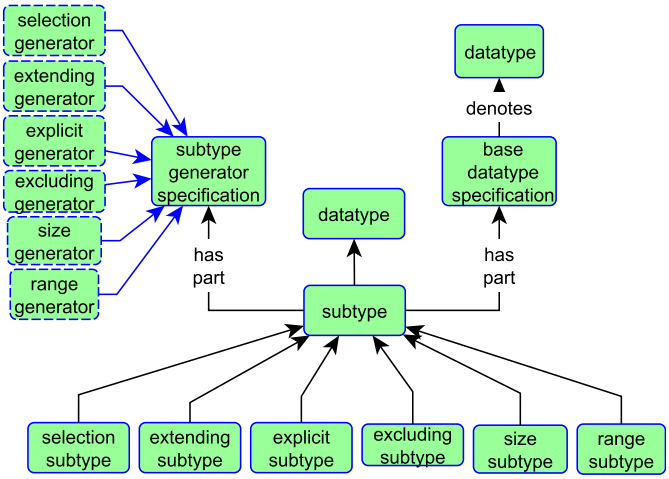
5.3.4 Subtype

A *subtype* IS-A *datatype* that is derived from an existing datatype by restricting the value space to a subset of the base datatype, while maintaining all operations. A subtype is defined by specifying a *subtype generator*, a *base datatype specification* and *value expressions* (See Figure 5.16(a)). The *subtype generator specification* specifies the relationship between the value spaces of the base type and the subtype. In OntoDT, we define the following instances of subtype generator: `selection generator`, `exclusion generator`, `extension generator`, `explicit subtype generator`, `size generator`, and `range generator` (see Figure 5.16(a)). The *base type specification* denotes the base datatype which is subtyped. The *value expressions* are *data items* that specify additional parameters for the subtype specification. Examples of value specifications include: *range specification*, *upper bound specification*, *lower bound specification* and others.

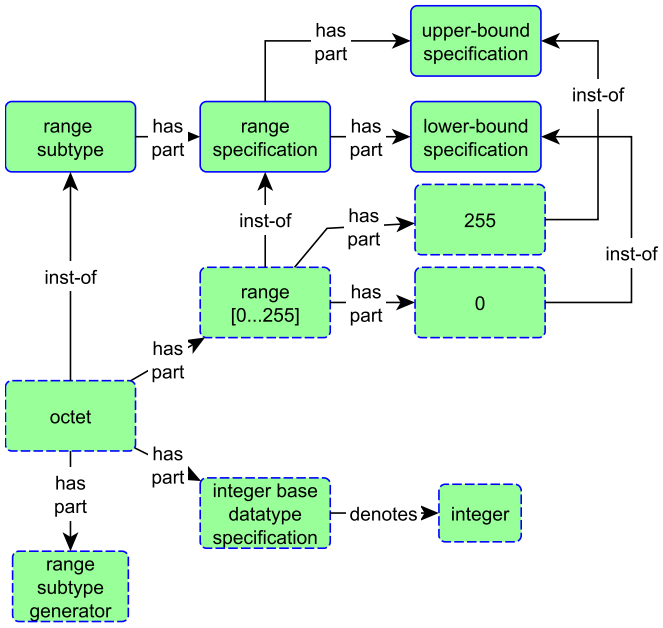
Each instance of the class subtype generator belongs to a separate class of datatype subtype from the following list:

- the *range subtype* is defined in OntoDT as a subtype of any ordered dataype; it is constructed by placing a new upper and/or lower bounds on the value space;
- the *selection subtype* is defined in OntoDT as a subtype of any exact datatype; it is specified by enumerating the values of the new value space;
- the *exclusion subtype* is defined in OntoDT as a subtype of any exact datatype, by enumerating the values which are being excluded from the new value space;
- the *size subtype* is defined in OntoDT as a subtype of any *bag*, *set*, *sequence*, or *table* datatype by specifying bounds on the number of elements of the base datatype;
- the *extension subtype* is defined in OntoDT as a subtype of any datatype, where the value space of the extended datatype contains the value space of the base datatype as a proper set; and
- the *explicit subtype* is defined in OntoDT as a subtype of a base datatype that includes a definition of a construction procedure for obtaining the subset value space of the new subtype.

In Figure 5.16(b), we present an example of a range subtype. A range subtype places a new upper and/or lower bound on the value space of an ordered base datatype. An *octet* is an instance of *range subtype*. It has as parts a *range subtype generator*, an *integer base datatype specification* denoting the base type integer, and *range specification*, specifying the range with the *upper bound specification* and the *lower bound specification*.



(a) class view



(b) instance view

Figure 5.16: The class subtype in OntoDT and example of an instance.

5.3.5 Defined datatype

A *defined datatype* is a *datatype* defined by a type specification. A *type specification* defines a new datatype that refers to a datatype or a datatype generator. This specification is used to rename an existing datatype, to define the syntax of a new datatype and to define the syntax of a new datatype generator. It includes a *type identifier*, a *type-parameter list* and a *type definition*. Examples of defined datatypes in OntoDT include the *labeled graph* class and its two subclasses *tree datatype* and *DAG datatype*[1].

[1]DAG=Directed Acyclic Graph

6 OntoDM-core: Ontology Module for Core Data Mining Entities

In this chapter, we present the OntoDM-core ontology module. First, we present the OntoDM-core goal, scope and competencies (Section 6.1). Next, we discuss the implementation the OntoDM-core ontology module (Section 6.2). Finally, we describe in more detail the structure of the ontology module (Section 6.3).

6.1 Goal, scope and competencies

In data mining, the data used for analysis are organized in the form of a dataset. Every dataset consists of data examples. The task of data mining is to produce some type of a generalization from a given dataset. Generalization is a broad term that denotes the output of a data mining algorithm. A data mining algorithm is an algorithm, that is implemented as computer program and is designed to solve a data mining task. Data mining algorithms are computer programs and when executed they take as input a dataset and give as output a generalization.

In this context, we introduce OntoDM-core as an ontology module for representing core data mining entities. The goal of this ontology module is to represent the core data mining entities, and be general enough to represent the mining of structured data. Based on the main aim of OntoDM-core and the ontology desiderata, we established a list of competency questions our ontology module is designed to answer. The most important competency questions are listed in Table 6.1. Judging from the list of questions, the ontology module will include information on datasets, types of datasets, data mining tasks, generalizations, parameter settings, data mining algorithms, implementation of algorithms, data mining software, and the processes of execution of algorithms and execution of generalization.

6.2 Implementation

In order to represent core data mining entities for structured data, we model the data mining process in the OntoDM-core ontology module by reusing and extending upper level classes from the OBI ontology, IAO ontology, the SWO ontology, and consequently the BFO ontology. In addition, the OntoDM-core module directly imports the OntoDT module, in order to allow the representation of datatypes for the purpose of describing the entities for mining structured data. Finally, the implementation of OntoDM-core is compliant with the design principles discussed in Chapter 4.

In the OntoDM-core ontology module, we distinguish three levels of description (see Section 4.4). In the remainder of this section, we present all three levels of description in context of OntoDM-core. The first level is the specification level (Section 6.2.1) and contains the information entities needed to describe and specify the entities participating in the data mining process. The second level is the implementation level (Section 6.2.2) and contains entities that are concretizations of the specification entities, and are realized in the

Table 6.1: OntoDM general competency questions.

Q_n	**Query**
1	Which datasets have data belonging to a datatype X?
2	Which data mining tasks can be formulated on data of datatype X?
3	Which data mining tasks can be formulated on a dataset X?
4	Which data mining algorithms solve a data mining task X?
5	Which data mining algorithms are applicable to data of datatype X?
6	What is the set of the possible generalization types that are given as output by solving a data mining task X on data of type Y?
7	What is the set of implementations for a DM algorithm X that have a parameter Y?
8	What is the set of all generated generalizations on a dataset X?
9	Which DM software toolkit contains an implementation of a DM algorithm X?
10	On which computer a specific data mining algorithm X has been executed?
11	What is the parameter setting of a given execution of a data mining algorithm X on a computer Y, having as input a dataset Z?
12	What is the set of datasets on which a given generalization X has been executed?

data mining process. The third level is the application level (Section 6.2.3) and contains processual entities in order to represent processes that occur in a data mining.

6.2.1 Specification level

The specification level of OntoDM-core (see Figure 6.1) consists of classes that are extension of IAO class *information content entity*. By importing the OntoDT module, the specification level of OntoDM-core also contains classes for characterizing datatypes, such as datatype, datatype generator, and others (see Section 5). At the top level, the specification level includes classes for representing data such as *data item*, *specification entitiy*, *representation*, and *directive information entity*. These top level classes are further sub-classed in order to provide representational mechanism to support the representation of mining structured data.

For example, in OntoDM-core we define a *specification entity* as an information entity that represents an explicit description of requirements to be satisfied by a material, product, or service. In a data mining context, we further define subclasses of specification entities such as *dataset specification*, *mapping specification*, *parameter specification*, *data specification*, *feature specification*, *feature set specification*, *generalization language specification*, and others. Another very important top level class is the *directive information entity* that represents specifications of a processes that can be concretized and realized. It includes subclasses for representing data mining tasks (subclass of *information processing objective*), algorithms, software, scenarios (subclass of *protocol*), and others.

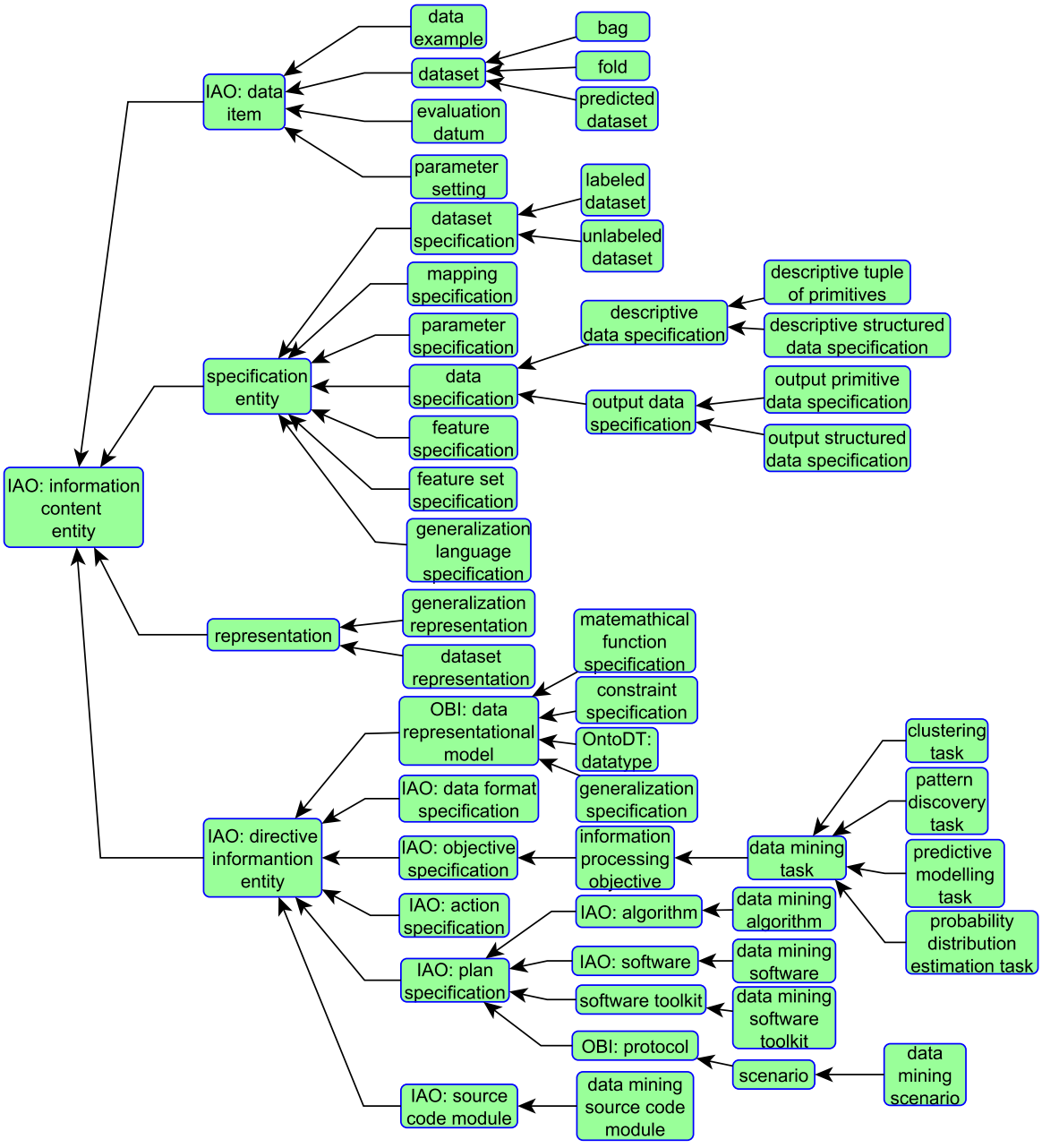


Figure 6.1: OntoDM-core specification level.

6.2.2 Implementation level

The implementation level of OntoDM-core (see Figure 6.2) consists of classes that are extension of the BFO class *specifically dependent continuant*. By importing the OntoDT module, the specification level of OntoDM-core also contains classes for characterizing qualities of datatypes, such as datatype quality, datatype generator quality, and others (see Chapter 5). At the top level, the implementation level includes classes for representing *realizable entities*, such as *roles*, *plans* and *generalizations*, and *qualities*.

Both top level classes are further sub-classed in order to provide a representational mechanism to support the representation of mining structured data. For example, we define *parameter* and *generalization quality* as subclasses of *quality*. Furthermore, we define *train set* and *test set* as dataset roles, being the subclasses of *role*. Finally, we define an *algorithm implementation* as a subclass of *plan*.

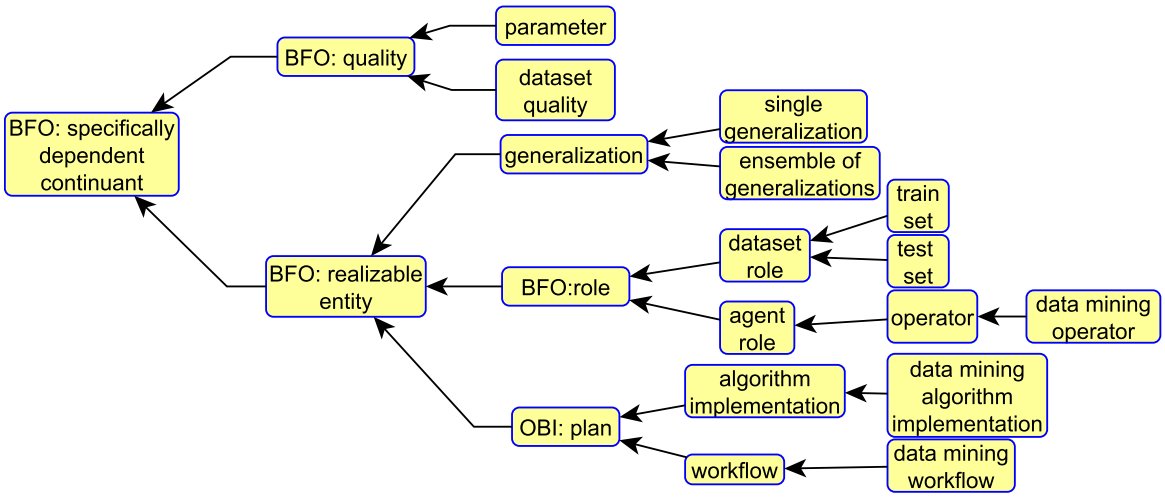


Figure 6.2: OntoDM-core implementation level.

6.2.3 Application level

The application level of OntoDM-core (see Figure 6.3) consists of classes that are extensions of OBI class *planned process*. At the top level it includes classes for representing *protocol execution*, *information processing*, and *validation*. The top level classes are further sub-classed in order to provide a representational mechanism to support the representation of processes for mining structured data. For example, we define both *algorithm execution* and *generalization execution* processes as subclasses of the *information processing* class.

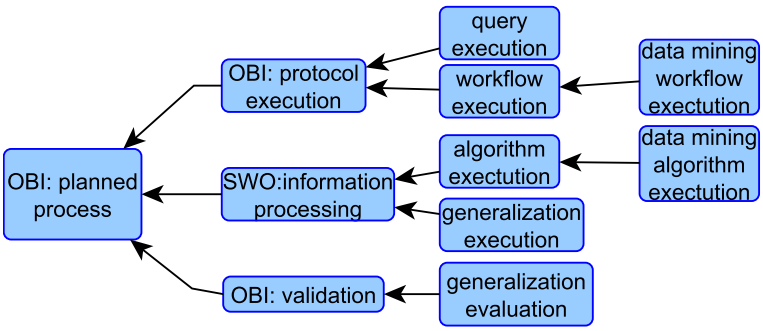


Figure 6.3: OntoDM-core application level.

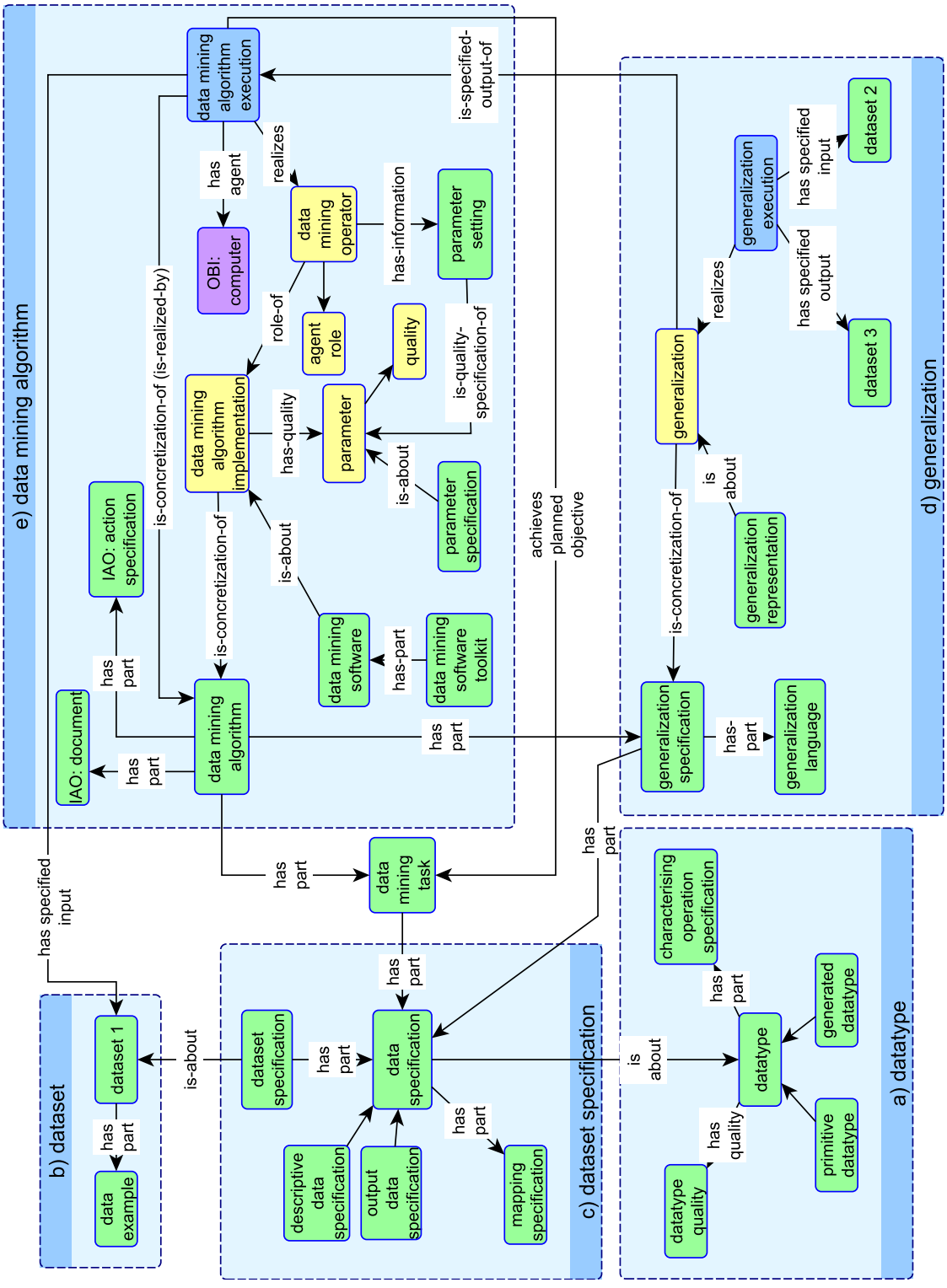


Figure 6.4: Key data mining classes and their relations. The boxes in green denote the information entities classes, the yellow boxes denote the specifically dependent continuants (realizable entities and qualities) and the blue boxes denote planned process classes. The violet boxes represent material entities.

6.3 Ontology module description

In this section, we present the representation of each of the core data mining entities for structured data and discuss the most important representational issues that came up in the process of modeling in the OntoDM-core ontology module. In Figure 6.4, we present the structure of the key OntoDM classes with the most important relations.

The entities are grouped into clusters depending on the major entity that they represent. First, we a have cluster of entities for representing datatypes (see Figure 6.4a), that is directly imported from the OntoDT module. Next, we have a cluster for representing dataset (see Figure 6.4b) and dataset specification (see Figure 6.4c). Furthermore, we have the data mining task and a cluster for representing knowledge about generalizations (see Figure 6.4d). Finally, we have a cluster for representing knowledge about data mining algorithms (see Figure 6.4e).

In the remainder of this section, we will survey the key classes having in mind the different description levels and the competency questions designed to answer (see Table 6.1). First, we present the dataset entity, including dataset specification, and introduce a taxonomy of datatypes (Section 6.3.1). Next, we focus on the data mining task entity and discuss the fundamental data mining tasks, and introducing a taxonomy of predictive modeling tasks (Section 6.3.2). In line with the tasks, we present the generalization entity, and we introduce a taxonomy of generalizations: More specifically we introduced a taxonomy of predictive models (Section 6.3.3). Furthermore, we discuss the representation of data mining algorithms in OntoDM-core, by introducing a three-level view on algorithms and a taxonomy of data mining algorithms (Section 6.3.4). Finally, we conclude this section by discussion how we represent constraints, constraint-based data mining tasks (Section 6.3.5), and data mining scenarios (Section 6.3.6).

6.3.1 Dataset

The main ingredient in the process of data mining is the dataset. In the IAO ontology, a dataset is defined as 'a data item that is an aggregate of other data items of the same type that have something in common'. We extend the IAO dataset class, by further specifying that a *DM dataset* HAS-PART *data examples* (See Figure 6.5). In addition, for the purpose of characterizing the data in the dataset, we define a *dataset specification* class that represents a specification entity that IS-ABOUT a *dataset*. In other words, the dataset specification is used to describe what is inside the dataset, for the purpose of automatic reasoning about datasets. For example, to characterize the type of data present in the dataset and its position, the dataset specification includes a *data specification*, which is an entity that IS-ABOUT a datatype.

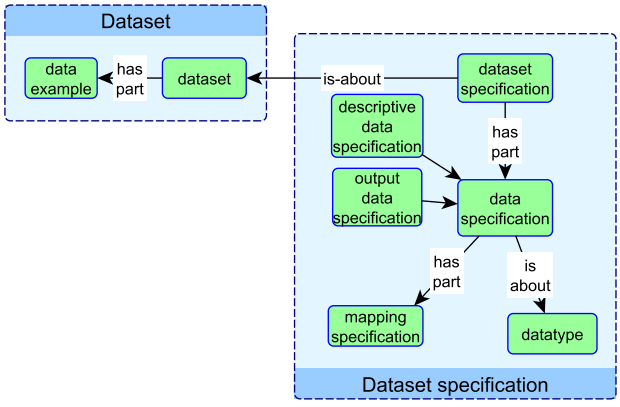


Figure 6.5: Representation of a dataset in OntoDM.

In the remainder of this section, we first discuss the data specification and focus on the datatypes that occur in the domain of data mining and discuss how we represent them using the language of OntoDT (Section 6.3.1.1). Next, we present the dataset specification and we propose a taxonomy of datasets based on data specifications (Section 6.3.1.2). Finally, we present an example of a dataset instance (Section 6.3.1.3).

6.3.1.1 Data specification and datatypes in data mining

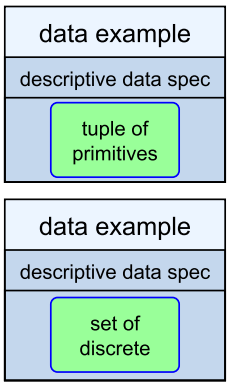
As we mentioned, a *data specification* IS-A *specification entity* that IS-ABOUT a *datatype*. In addition, it specifies on which part of the data example the *datatype* applies to, by including the *mapping specification* entity as its part (see Figure 6.5). For example, we distinguish between parts of data examples used just for descriptive purposes (e.g., in the case of clustering) named descriptive part, and output parts used for predictive purposes (e.g., in the case of predictive modeling) named output part. In this setting, we differentiate between two data specifications: descriptive data specification and output data specification. A *descriptive data specification* specifies the data used for descriptive purposes. *Output data specification*, on the other hand, specifies the data used for output purposes.

Since data specifications specify the datatype of the data by including the information about the datatype, in the OntoDM-core module we import the mechanism for representing arbitrary complex datatype from OntoDT module (see Chapter 5). This is necessary in oder to represent data mining specific datatypes, and to allow the characterization the data mining datasets for the purpose of automatic reasoning about datasets and their classification into a taxonomy of datasets. Examples of datatypes frequently used in data mining include tuple of primitive datatypes, set of discrete, sequence of reals, sequence of characters, bag of words, and others. Finally, in the context of mining structured data, the data used for data mining can have arbitrarily complex structure, and OntoDM ontology is the only one that can successfully support characterization of data for describing the process of mining structured data.

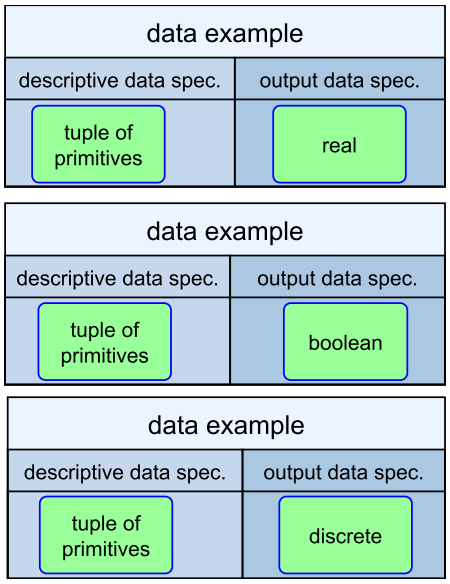
In Figure 6.6, we present examples of data specifications and datatypes in data mining. First, in Figure 6.6(a), we present characterization of data examples that have only descriptive data specification. These include the feature-based data examples, that are characterized by having tuple of primitives as their datatype, and transactional data examples, that have a set of discrete as the underlying datatype. Next, in Figure 6.6(b), we have data examples most often used in traditional predictive data mining. These include data examples that have a tuple of primitives as a descriptive data specification and a primitive datatype (e.g., boolean, real, discrete) as an output data specification. Furthermore, in Figure 6.6(c) we present examples of feature-based data examples that have tuple of primitives at the output. Finally, in Figure 6.6(d), we present examples of feature-based data examples that have structured output. These data examples have structured datatypes as their output data specification (e.g., set of discrete, sequence of real, tree/DAG with boolean edges and discrete nodes). All these types of data examples are used in real life data mining applications and will be elaborated in more details in the remaining of this Chapter.

6.3.1.2 Dataset specification and taxonomy of datasets

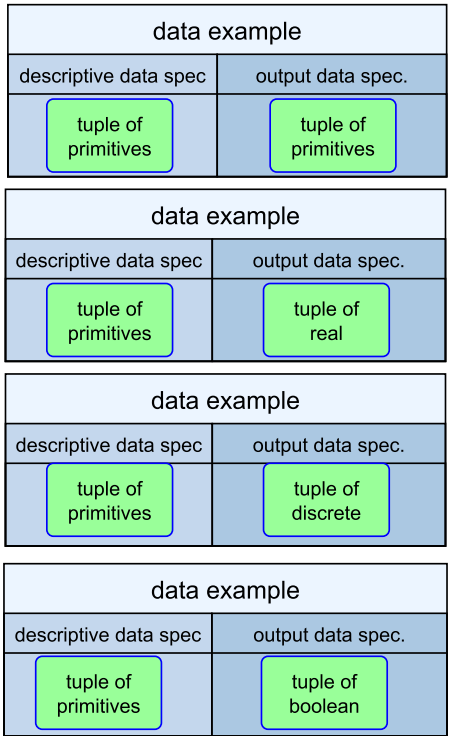
Unlike the *data specification*, that represents an entity for characterizing the datatype in the data examples, *dataset specification* specifies the type of the dataset based on the type of data it contains. It includes *data specifications* as its part. In addition, depending on the type of dataset, a dataset specification can include as parts a number of other entities. For example, the data specification for feature-based labeled dataset contains the feature specifications (such as name, ID, etc) of the features (attributes) included in the tuple.



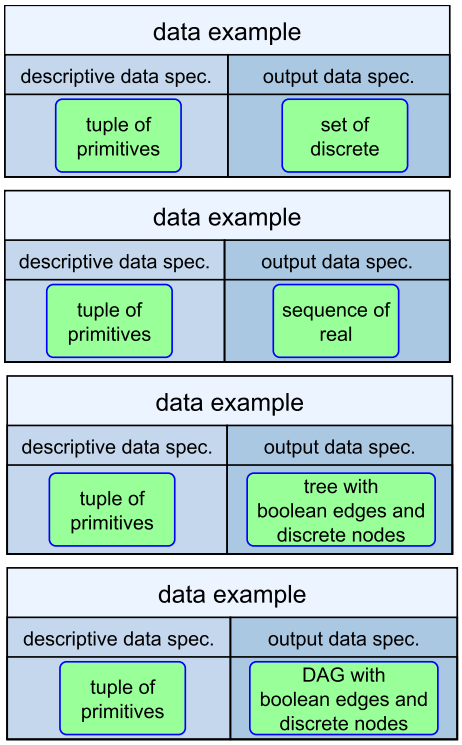
(a) feature-based data example and transactional data example



(b) feature-based data examples with primitive output



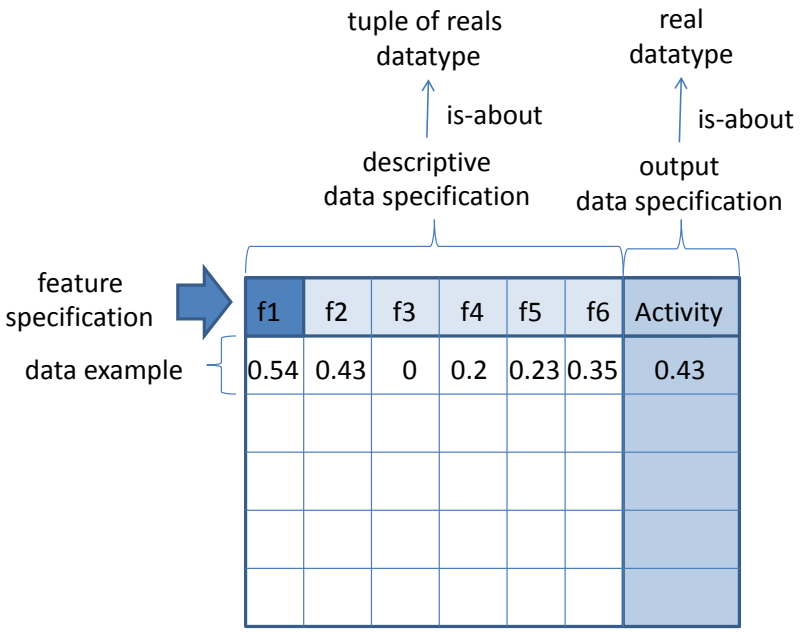
(c) feature-based data examples with tuple of primitives as output



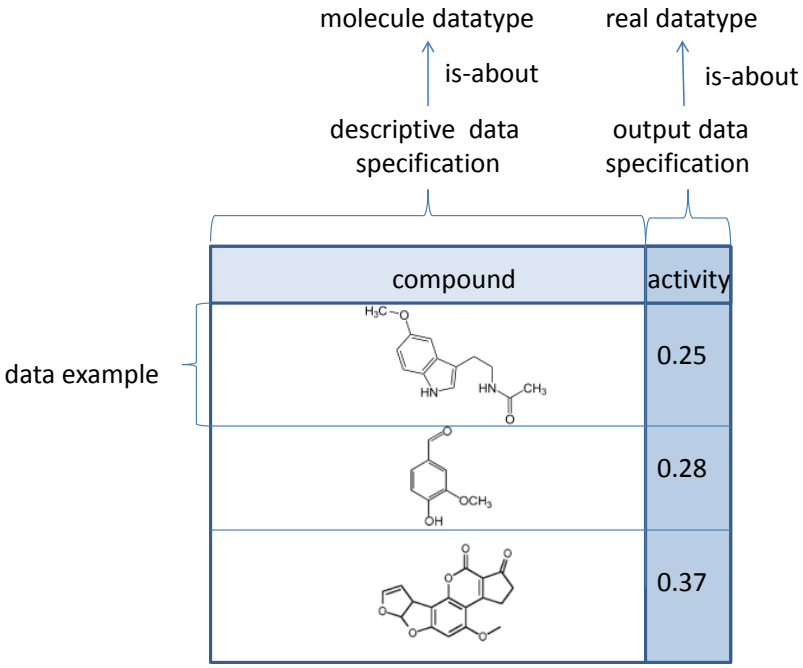
(d) feature-based data example with structured output

Figure 6.6: Data specifications and datatypes in data mining.

For example, in Figure 6.7 we present two different datasets and their dataset specifications. The dataset in Figure 6.7(a), represents a featured-based labeled dataset that is characterized by descriptive data specification that specifies that the descriptive part is tuple of reals, and output data specification that specifies that the output part is real datatype. The dataset in Figure 6.7(b), represents a molecular dataset that is characterized by descriptive data specification that specifies that the descriptive part is molecular datatype (defined as an extension of defined datatype class), and output data specification that specifies that the output part is real datatype (such as the one presented in Figure 6.7(a)).



(a) feature-based labeled dataset



(b) structured-based labeled dataset

Figure 6.7: Examples of dataset specifications.

Using data specifications, and indirectly the taxonomy of datatypes from the OntoDT module, we propose a taxonomy of datasets (see Figure 6.8). At the first level of the taxonomy of datasets, we have *unlabeled dataset* and *labeled dataset*. An *unlabeled dataset* IS-A *dataset specification* for datasets that have only a *descriptive data specification*. These datasets are most often used for the task of clustering. A *labeled dataset* IS-A *dataset specification* that also has an *output data specification*. They are usually used for predictive modeling.

Unlabeled dataset. For the unlabeled dataset, at the second level (see Figure 6.8), we distinguish between *feature-based labeled dataset* and *structure-based unlabeled datasets*. Feature-based unlabeled dataset IS-A unlabeled dataset that has tuple of primitives as the underlying descriptive datatype. Structure-based unlabeled dataset IS-A unlabeled dataset that has some structured datatype (other than tuple of primitives) as the underlying descriptive datatype.

Labeled dataset. For the labeled dataset, at the second level (see Figure 6.8), we distinguish between *labeled dataset with primitive output* and *labeled dataset with structured output*. The first type of dataset has as output data specification any primitive datatype, while the second type of dataset has as output data specification a structured datatype. Both dataset types can have an arbitrary datatype as a descriptive data specification.

If we focus next only on labeled datasets, the taxonomy can be further extended based on the descriptive and output specifications. Labeled dataset with primitive output is extended with *feature-based labeled dataset with primitive output* and *structure-based labeled dataset with primitive output*, where the first sub-class has a tuple of primitives as a descriptive specifications, while the second sub-class has any structured datatype (other than tuple of primitives) as a descriptive. Feature-based labeled datasets with primitive output are further extended based on the type of primitive output. This includes sub-classes such as *regression dataset* (having as output datatype real), *binary classification dataset* (having as output datatype boolean), and *multi-class classification dataset* (having as output datatype discrete).

In analogy, labeled dataset with structured output is extended with *feature-based labeled dataset with structured output* and *structure-based labeled dataset with structured output*, where the first sub-class has a tuple of primitives as a descriptive specifications, while the second sub-class has any structured datatype (other than tuple of primitives) as a descriptive. Feature-based labeled datasets with structured output are further extended based on the type of structured output. This includes sub-classes such as *multi-target prediction dataset* (having as output datatype a tuple of primitives), *multi-label classification dataset* (having as output datatype a set of discrete), *feature-based time series prediction dataset* (having as output datatype a sequence of reals), and *hierarchical classification dataset* (having as output datatype a labeled graph with boolean edges and discrete nodes). Finally, *multi-target prediction dataset* is further extended depending on the primitive datatypes that compose the tuple. This includes the following sub-classes: *multi-target regression dataset*, *multi-target binary dataset*, and *multi-target multi-class dataset*.

6.3.1.3 An example of a dataset instance: The EDM dataset

We present an example representation of a dataset instance by looking in more detail at a real life EDM dataset (Karalic and Bratko, 1997) from the literature (see Fig. 6.9). Electrical Discharge Machining (EDM) is a machining method primarily used for hard materials that are electrically conductive. The workpiece surface is machined by electrical discharges occurring in the gap between two electrodes, the tool and the workpiece. The gap is continuously flushed by the dielectric fluid.

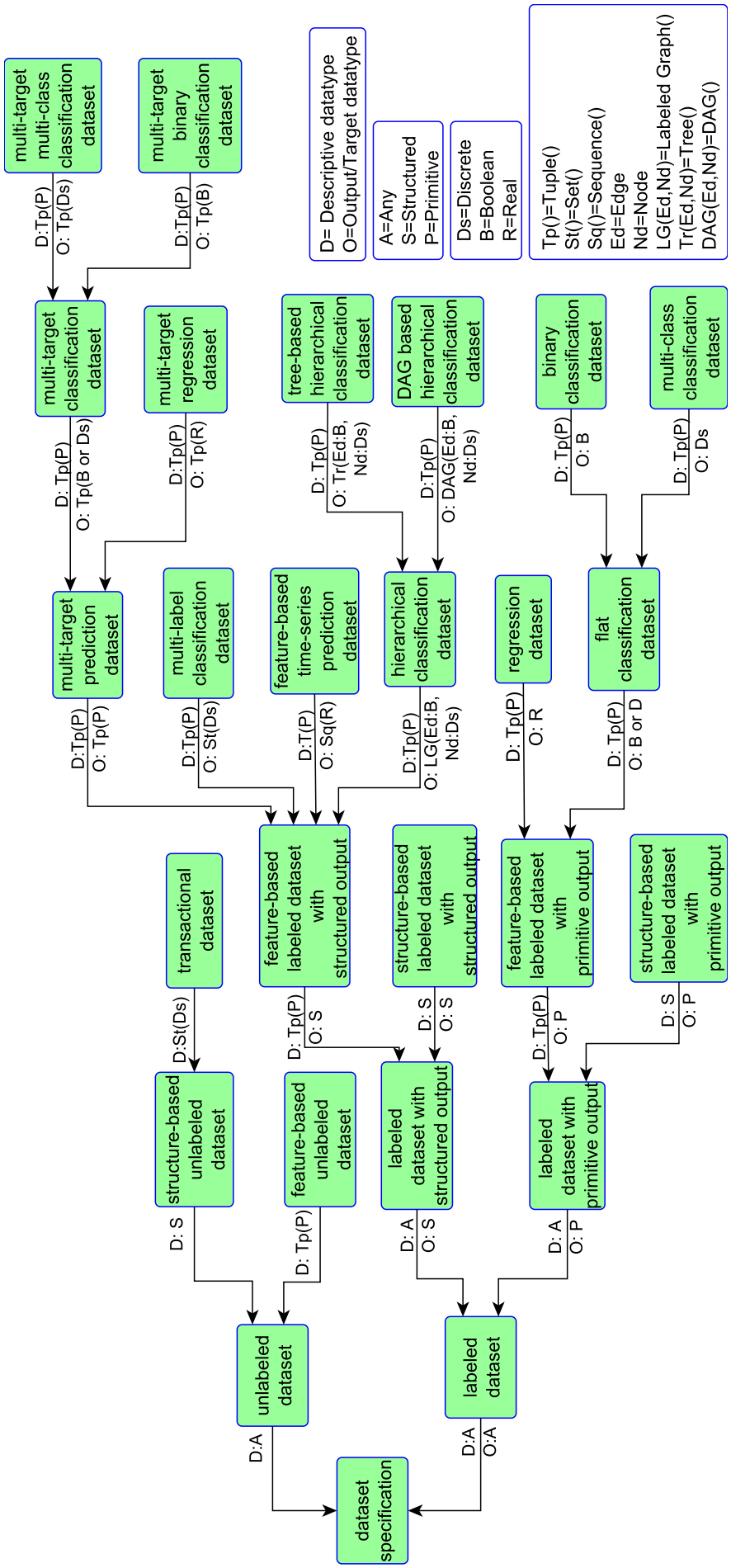


Figure 6.8: Taxonomy of datasets. The full line arrows represent IS-A relations. The labels on the arrows are descriptive and output data specifications used as criteria for building the taxonomy.

The process consists of numerous randomly ignited mono-discharges generating a crater-textured surface. The stability and quality of the process depend on many parameters such as the workpiece material, dielectric fluid type, size of the gap between the electrodes, flow of the dielectric fluid, electric voltage between the electrodes, electric current between the electrodes, characteristics of discharges, and others. While these parameters have some predefined values, machining time can often be shortened by their dynamic control. A human operator is normally employed for this purpose.

The data mining task at hand is to learn automatically to reproduce the operator's behavior. The dataset describes 154 actions taken by the operator where he controlled two variables (output features), the flow and the gap. For each variable, three actions were possible: he increased the variable (numerical value 1), decreased the variable (value -1), or took no action (value 0). The reasons for each action are described in terms of 16 numeric features which define the type and size of electrical pulses and their history (e.g., mean value and standard deviation of effective pulses within the last 5 seconds).

In OntoDM-core, we represent the `EDM dataset` as an instance of the *dataset* class (see Fig. 6.9). Tha `EDM dataset` HAS-PART `EDM data examples`. In order to further characterize the dataset, we introduce the `EDM dataset specification`, that is an instance of the *multi-target multi-class dataset* type (see the taxonomy of datasets in Fig. 6.8), because the dataset has a tuple of primitive datatypes on the descriptive side (16 numeric features) and a tuple of discrete datatype on the output side (2 discrete features).

The `EDM dataset specification` instance further includes data specifications on the descriptive and the output parts: `EDM descriptive tuple of primitive` and `EDM output tuple of discrete`. Each of the specifications include the description of the concrete datatype instance that is applicable to the data at hand and the description of the feature set. For example, the `EDM descriptive tuple od primitives` IS-ABOUT an `EDM tuple of primitives datatype`. According to the OntoDT ontology module, this datatype is an instance of the OntoDT *tuple of primitives* class, and it has as part a `tuple generator` and a *field component list* containing specifications of all members of the tuple, denoting for each member its datatype and identifier.

The *feature set specification* is a specification entity that has as parts *feature specifications* of all features in the dataset, such as names, statistical properties and others. For example, we have `EDM descriptive feature set specification` that is an instance of the *feature set specification* class. It contains as parts 16 `EDM des_feat xx` that are instances of *feature specification* class. Each feature specification instance has assigned name. For example, `EDM des_feat 1` HAS-NAME `'ASM_A_MeanT'`. In similar way, we represent also the EDM output dataset specification.

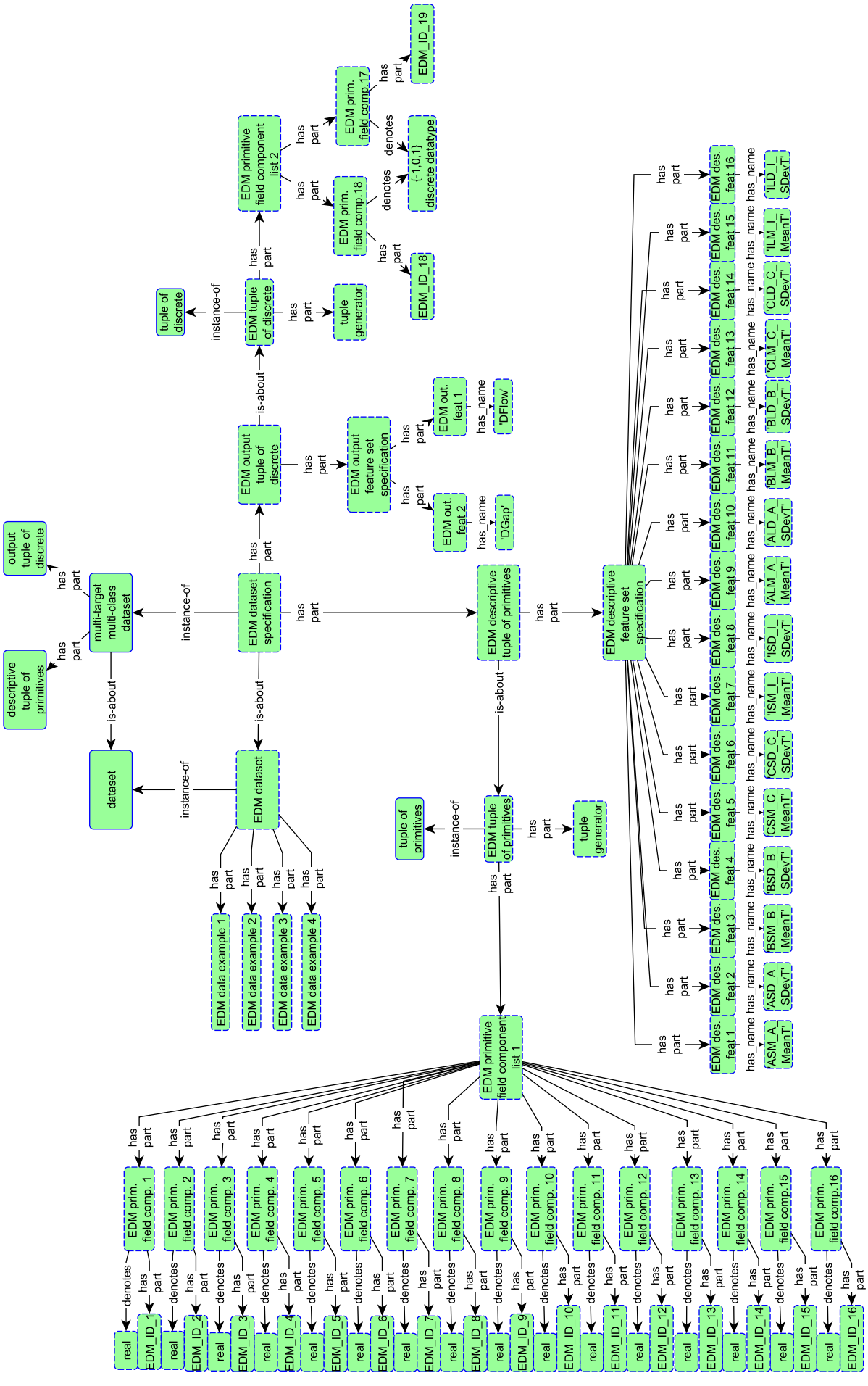


Figure 6.9: Example of representation of a dataset in OntoDM-core.

6.3.2 Data mining task

The task of data mining is to produce a generalization from a given dataset. In OntoDM, we define a *data mining task* as sub-class of IAO *objective specification*. The *data mining task* specifies the objective that a *data mining algorithm execution* process needs to achieve when executed on a *dataset* in order to produce as output a *generalization* (see Fig. 6.10).

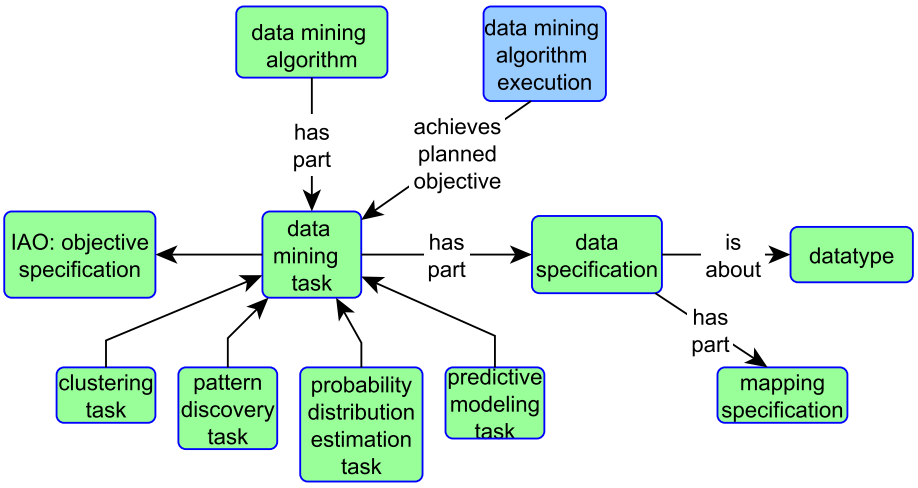


Figure 6.10: Data mining task in OntoDM-core.

A data mining task is an information entity that has as part data specification (see Figure 6.10). That means that an instance of a data mining task is directly dependend on the type of data. According to Džeroski (2007), in OntoDM-core we distinguish between four basic clases of data mining tasks based on the generalizations that are produced as output: a *clustering task*, a *pattern discovery task*, a *probability distribution estimation task*, and a *predictive modeling task*.

The classes of tasks are fundamental and they are defined on an arbitrary type of data (by using the descriptive data specification), except for the predictive modeling task that is defined on a pair of datatypes (it has both descriptive and output data specification). For example, for the task of *predictive modeling* the objective is to find a mapping from the description to the output, in the form of a *predictive model*, given a labeled dataset that consists of data examples that have a defined descriptive and output data specifications.

In the remainder of this section we first propose a taxonomy of data mining tasks and focus on on the fundamental data mining tasks (6.3.2.1). Next, we extend the initial taxonomy of tasks and propose a more in depth taxonomy of predictive modeling tasks using data specifications (Section 6.3.2.2). Finally, we present an example of how the taxonomy can be further extended to cover very specific tasks, as is the case for the task of hierarchical classification (Section 6.3.2.3).

6.3.2.1 Taxonomy of fundamental data mining tasks

From the definition of data mining task we can see that the definition of a data mining task is directly dependent on the data specification, and indirectly depends on the datatype on the data at hand. This allows us to form a taxonomy of data mining tasks having as criteria the type of data. At the first level of the taxonomy of data mining tasks (see Figure 6.10) we have the fundamental data mining tasks as defined by Džeroski (2007). Furthermore, as mentioned in the introduction of this section, all fundamental data mining tasks have as part descriptive data specification, and only predictive modeling has as part of its definition output data specification. This means that at the next levels of the taxonomy of data mining task will exclusively depend on the datatype of the descriptive data, and in the case

of predictive modeling task it will depend also on the datatype of the output data. In the remainder of this section we focus on the definition of the fundamental tasks at level one of the taxonomy.

Clustering task The task of clustering in general is concerned with grouping objects into classes of similar objects (Kaufman and Rousseeuw, 1990). Given a set of examples organized in a unlabeled dataset having a data specification that denotes the datatype T, the task of clustering is to partition these examples into subsets, called clusters. The final objective of clustering is to achieve high similarity between examples within individual clusters (intra-cluster similarity) and low similarity between objects that belong to different clusters (inter-cluster similarity).

Pattern discovery task The task of pattern discovery is to find all local patterns from a given pattern language that satisfy the required conditions. A prototypical instantiation of this task is the task of finding frequent itemsets (sets of items, such as $\{bread, butter\}$), which are often found together in a transaction (e.g., a market basket) (Agrawal et al., 1993). With the increasing amount of interest in mining complex data, mining frequent patterns is also considered for structured data. This includes tasks such as mining frequent subgraphs in a sequence of graph data or mining frequent subsequences.

Probability distribution estimation task A dataset having a data specification denoted with the type T is assumed to be a sample of some population following a probability distribution. Every example from the dataset is assigned a non-negative probability (or density) by the probability distribution function (or density function). The most general data mining task is the task of estimating the (joint) probability distribution D over type T from a set of data examples or a sample drawn from that distribution (Hand et al., 2001).

Predictive modeling task In the task of predictive modeling, we are given a dataset that consists of examples of the form (d, o), where d is descriptive specification denoting datatype T_d and o is an output specification denoting datatype T_c. To learn a predictive model means to find a mapping from the description to the output, $m :: T_d \rightarrow T_o$, that fits the data closely. This means that the observed target values and the target values predicted by the model. The task of learning a predictive model can be seen as a special case of the task of estimating the probability distribution, as it can be formulated from the joint probability distribution $P((d, o))$ by deriving the conditional distribution $P(o \mid d)$.

6.3.2.2 Taxonomy of predictive modeling tasks

In the previous section, we introduced the task of *predictive modeling*. For this task we are given a labeled dataset that consists of data examples that have a descriptive and output part, both defined by the data specifications. The task of predictive modeling is to find a mapping from the description to the output, in a form of a predictive model. In this section, we illustrate the OntoDM-core mechanism that supports the representation of data mining tasks for structured data with a proposal for a taxonomy of predictive modeling tasks (see Figure 6.11). Using the output data specification as a criterion, at the first level we distinguish between the *primitive output prediction task* and the *structured output prediction task* (examples of such tasks can be found in Bakir et al. (2007)). In the first case, the output datatype is primitive, such as discrete, boolean or real; in the second case, it is some structured datatype, such as tuple, set, sequence or graph.

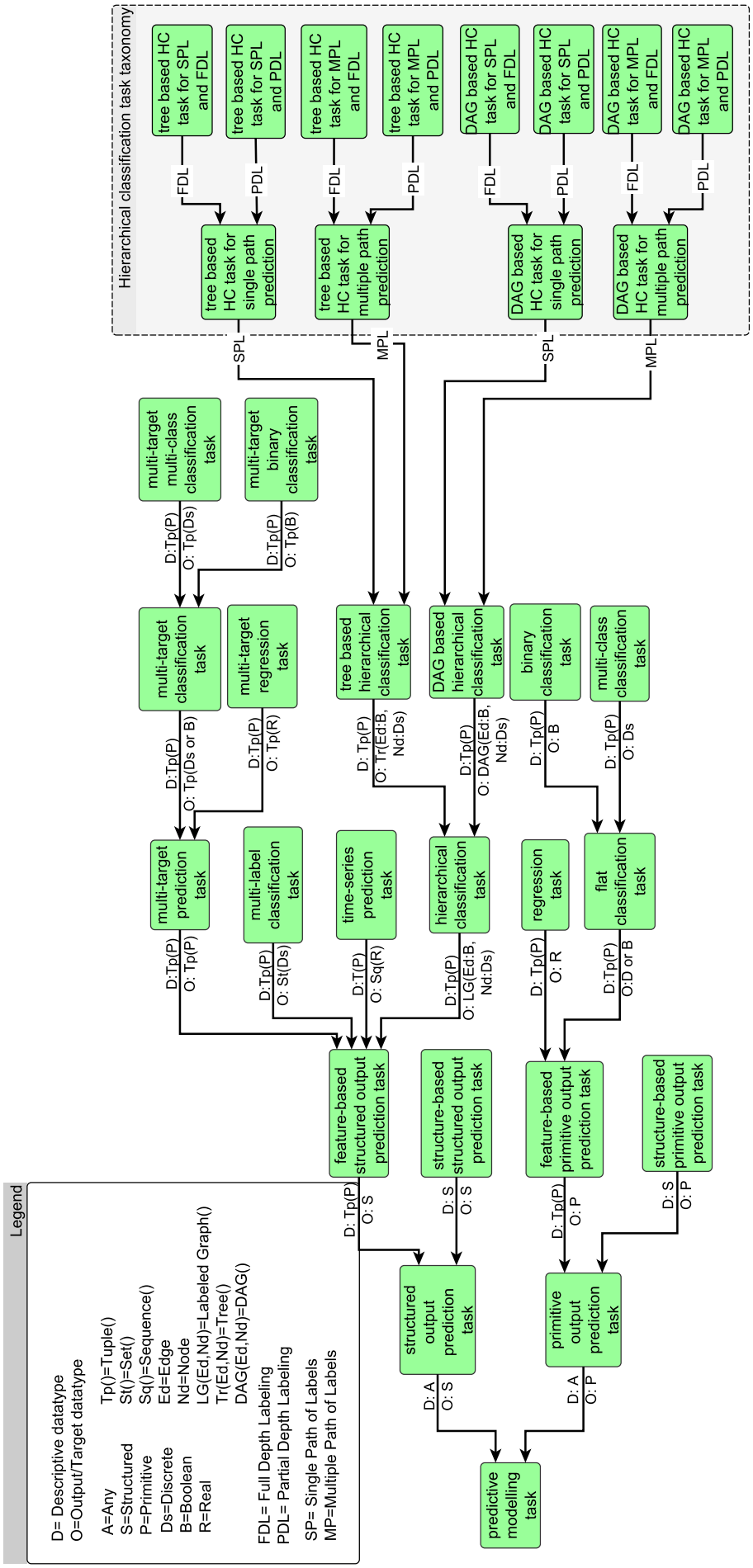


Figure 6.11: Taxonomy of predictive modeling tasks. The full line arrows represent IS-A relations. The labels on the arrows are descriptive and output data specifications used as criteria for building the taxonomy.

Primitive output prediction tasks. The *primitive output prediction tasks* can be feature-based or structure-based, depending on the datatype on the descriptive part. *Feature-based primitive output prediction tasks* are tasks that operate on data that have a tuple of primitives (set of primitive features) on the description side and a primitive datatype on the output side. This is the most exploited data mining task in traditional single-table data mining, described in all major data mining textbooks (e.g., Hand et al. (2001)). If we specify the output datatype in more detail, we can represent the *binary classification task*, the *multi-class classification task* and *regression task*; where the output datatype is boolean, discrete or real, respectively. *Stucture-based primitive output prediction tasks* are tasks that operate on data that have some structured datatype (other than tuple of primitives) on the description side and a primitive datatype on the output side.

Structured output prediction tasks. In similar way, *structured output prediction tasks* can be feature-based or structure-based. *Feature-based structured output prediction tasks* are tasks that operate on data that have a tuple of primitives (set of primitive features) on the description side and a structured datatype on the output side. *Structure-based structured output prediction tasks* are tasks that operate on data that have some structured datatype, other than tuple of primitives, on the description side and a structured datatype on the output side.

If we focus just on feature-based tasks and further specify a structured output datatype, we can represent a variety of structured output prediction tasks. For example, we can represent the following tasks: *multi-target prediction task* (Caruana, 1997) (which has as output datatype *tuple of primitives*), *multi-label classification task* (Tsoumakas and Katakis, 2007) (which has as output datatype *set of discrete*), *time-series prediction task* (Slavkov et al., 2010) (which has as output datatype *sequence of real*) and *hierarchical classification task* (Silla and Freitas, 2011) (which has as output datatype *labeled graph with boolean edges and discrete nodes*). Furthermore, a *multi-target prediction task* can be further specified into tasks such as: *multi-target binary classification task*, *multi-target multi-class classification task* (Demšar et al., 2006), and *multi-target regression task* (Kocev et al., 2009).

6.3.2.3 Further extensions: Taxonomy of hierarchical classification tasks

The taxonomy of predictive modeling tasks, discussed in the previous section, can be further extended on the basis of other criteria. These criteria are established by the needs of the sub-domains of data mining and the communities that are addressing the different (sub)tasks and developing algorithms to solve them. The sub-domains include text mining (Feldman and Sanger, 2006), mining graphs, trees (Cook and Holder, 2006) and sequences (Dong, 2009)). In this section, we present a proposal how we can further extend the taxonomy of hierarchical classification tasks by using the criteria presented by Silla and Freitas (2011), in their review paper on hierarchical classification across different application domains (see the upper part of Figure 6.11).

Silla and Freitas (2011), in their review paper, present a set of criteria for the classification of hierarchical classification (HC) tasks. The criteria include: the type of graph representing the hierarchical classes (tree or directed acyclic graph (DAG)), information whether a data example is allowed to have class labels associated with a single (SPL) or multiple paths (MPL) in the class hierarchy, and the information about the label depth of the data examples that can be full depth (FDL) or partial depth(PDL).

In line with our design principles, we represented their proposal in OntoDM-core module as a sub-taxonomy of the HC task (see the right hand side of Figure 6.11). At the first level of the sub-taxonomy, we distinguish between HC tasks that are based on discrete class labels organized in a tree structure and the ones organized in the form of a DAG, the first being denoted with the output data specification with tree datatype with boolean

edges and discrete nodes and the second one being denoted with the output specification with DAG datatype with boolean edges and discrete nodes. Next, at the second level, we distinguish between single (SPL) or multiple path labels (MPL). At the last level, we distinguish between tasks that have full (FDL) or partial labeling (PDL). For example, Vens et al. (2008) proposes an algorithm for learning decision trees for hierarchical multi-label classification which is in fact solving the task of hierarchical classification having a DAG as a output datatype, allowing multiple paths and partial depth labeling, represented in OntoDM-core as the *DAG based HC task for MPL and PDL*.

6.3.3 Generalizations

Generalization is a broad term that denotes the outcome of applying a data mining task (for a given dataset). Many different types of generalizations have been considered in the data mining literature. The most fundamental types of generalizations are in line with the data mining tasks outlined in Section 6.3.2. These include probability distributions, patterns, predictive models and clusterings.

In OntoDM-core, we consider and model three different aspects of generalizations and each of these aspects is aligned with a different description level (see Figure 6.12). This includes a specification of the generalization, a generalization as a realizable entity, and the process of executing a generalization. In the remainder of this subsection, we will focus in brief on each of the three aspects of generalizations. First, we present the generalization as a specification and present our proposal for a taxonomy of generalizations based on the fundamental generalization types, data specification, and the generalization language (Section 6.3.3.1). Next, we present in more details the taxonomy of predictive models (Section 6.3.3.2). Finally, we discuss the generalization as an implementation and the process of executing a generalization (Section 6.3.3.3).

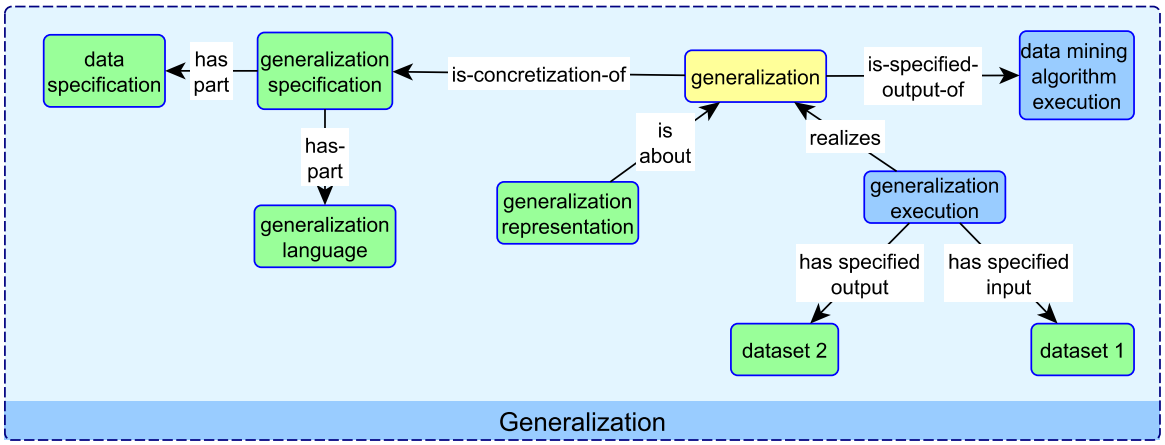


Figure 6.12: Representation of generalization in OntoDM-core.

6.3.3.1 Generalization specification and taxonomy of generalizations

In OntoDM-core, we define *generalization specification* class as a OBI *data representational model* that specifies the type of the generalization (see Figure 6.12). It includes as part *data specification* that implicitly takes into account the types of data that can be used to produce the generalization, and the *generalization language specification* that specifies the language in which the generalization is expressed. Examples of language formalisms for the case of a *predictive model* include the languages of: trees, rules, Bayesian networks, graphical models, neural networks, etc.

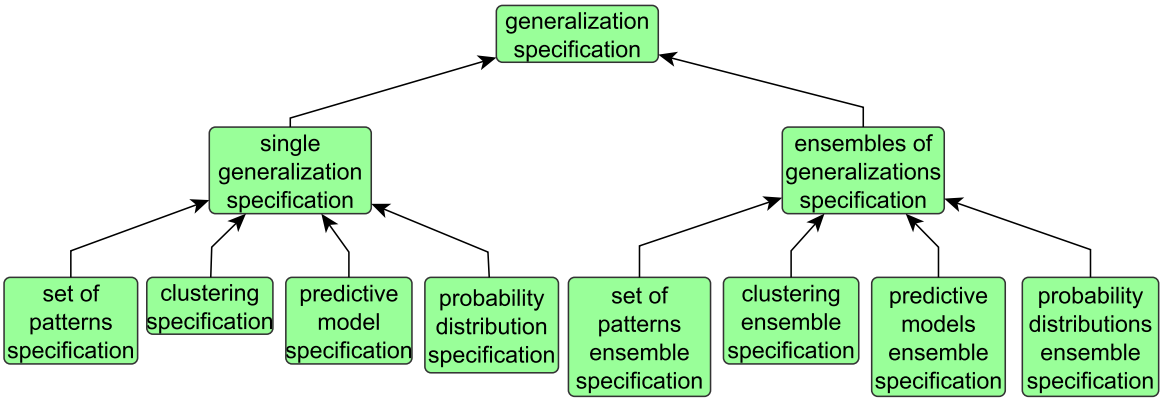


Figure 6.13: Taxonomy of generalizations in OntoDM-core.

As in the case of datasets and data mining tasks, we can construct a taxonomy of generalizations (see Figure 6.13). In OntoDM-core, at the first level, we distinguish between a *single generalization specification* and an *ensemble specification*. A *single generalization specification* denotes a generalization type produced by executing a *single generalization data mining algorithm*, such as a decision tree algorithm or a SVM algorithm. An *ensemble specification* denotes a generalization type produced by executing an *ensemble algorithm*, such as bagging, boosting, random subspaces, random forests etc. Ensembles of generalizations have as parts single generalizations.

At the second level, we distinguish between four core types of generalizations covering the four data mining tasks (see Section 6.3.3.1). This includes *set of patterns specification*, *clustering specification*, *predictive model specification*, and *probability distribution specification*, as sub-classes of *single generalization specification*, and *set of patterns ensemble specification*, *clustering ensemble specification*, *predictive model ensemble specification*, and *probability distribution ensemble specification*, as a sub-class of *ensemble specification*.

The taxonomy of generalizations can be further extended using the data specifications and generalization language as criteria. For example, in the OntoDM-core taxonomy, we can represent a *multi-target regression tree*, that is a *multi-target predictive model*: It has a tuple of primitives datatype at the description and a tuple of reals at the output part, and is expressed in the language of decision trees.

6.3.3.2 Taxonomy of predictive models

In Figure 6.14, we present how the taxonomy of generalizations can be extended for the case of predictive models. Because the generalization specification is strongly dependent on the data specification, the structure of the taxonomy of generalizations follows the structure of taxonomy of datasets and taxonomy of data mining tasks, presented in Section 6.3.1.2 and 6.3.2.2 respectively.

By using the output data specification as a criterion, at the first level we distinguish between the *predictive models for primitive output specification* and the *predictive models for structured output specification* (examples of such predictive models can be found in Bakir et al. (2007)). In the first case, the output datatype is primitive, such as discrete, boolean or real; in the second case, it is some structured datatype, such as tuple, set, sequence or graph.

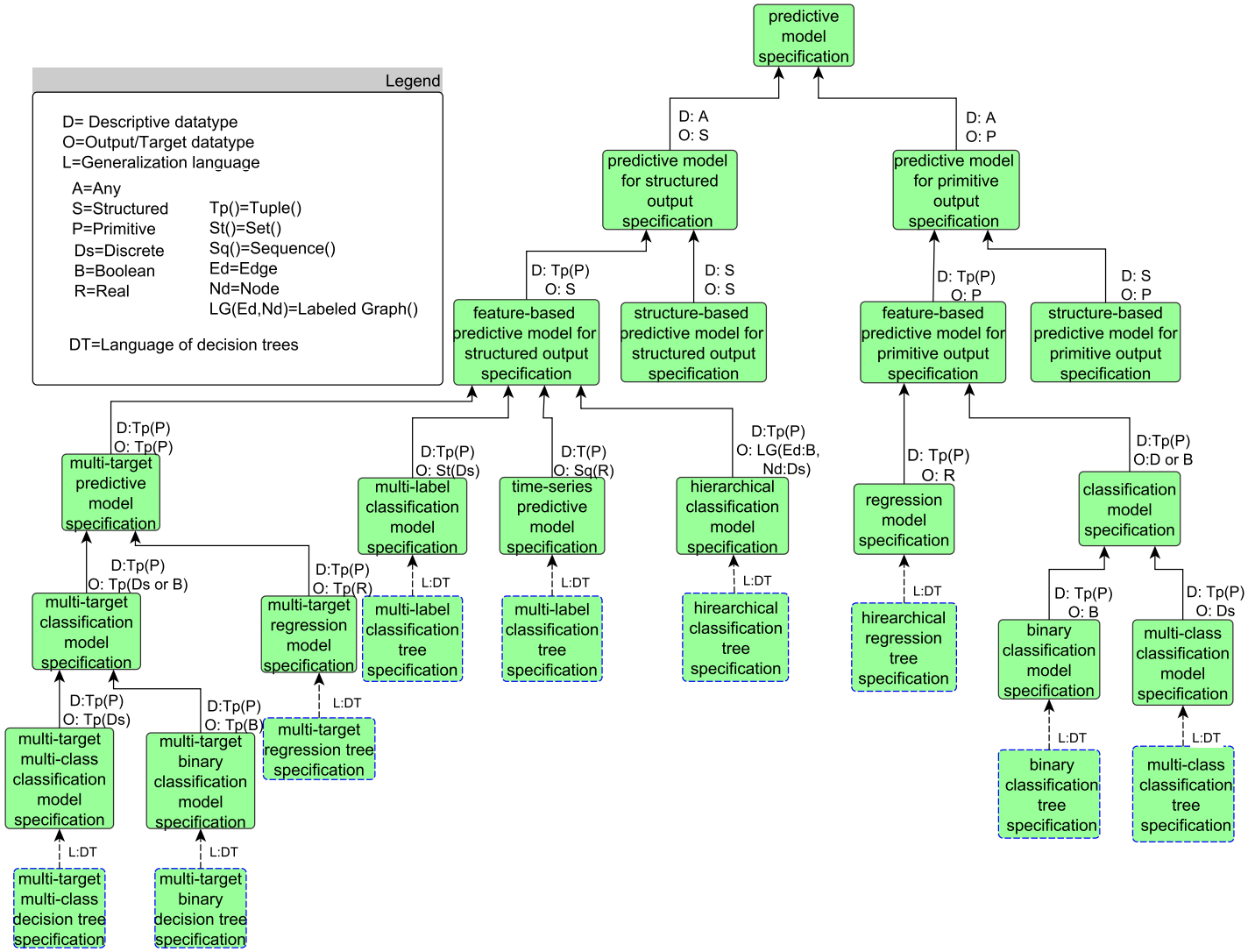


Figure 6.14: Taxonomy of predictive models in OntoDM-core with an example instances by having language of decision trees as generalization langugage. Full lines denote IS-A relation, while dashed lines denote INSTANCE-OF relation.

Predictive models for primitive output. The *predictive models for primitive output specification* can be feature-based or structure-based, depending on the datatype on the descriptive part. *Feature-based predictive model for primitive output specification* specify a model that is built on data that have a tuple of primitives (set of primitive features) on the description side and a primitive datatype on the output side. This is the most exploited type of models in traditional single-table data mining, described in all major data mining textbooks (e.g., Hand et al. (2001)). If we specify the output datatype in more detail, we can represent other classes of models such as the *binary classification model specification*, the *multi-class classification model specification* and *regression model specification*; where the output datatype is boolean, discrete or real, respectively. *Structure-based predictive model for primitive output specification* specifies class of models that operate on data that have some structured datatype (other than tuple of primitives) on the description side and a primitive datatype on the output side.

Predictive models for structured output. In similar way, *predictive models for structured output specification* can be feature-based or structure-based. *Feature-based predictive models for structured output specification* specify a class of models operate on data that have a tuple of primitives (set of primitive features) on the description side and a structured datatype on the output side. *Structure-based predictive model for structured output specification* specify class of models operate on data that have some structured datatype, other than tuple of primitives, on the description side and a structured datatype on the output side.

If we focus just on feature-based models and further specify a structured output datatype, we can represent a variety of structured output predictive models. For example, we can represent the following classes of types of models: *multi-target predictive model specification* (Caruana, 1997) (which has as output datatype *tuple of primitives*), *multi-label classification model specification* (Tsoumakas and Katakis, 2007) (which has as output datatype *set of discrete*), *time-series predictive model specification* (Slavkov et al., 2010) (which has as output datatype *sequence of real*) and *hierarchical classification model specification* (Silla and Freitas, 2011) (which has as output datatype *labeled graph with boolean edges and discrete nodes*). Furthermore, a *multi-target predictive model specification* can be further specified into models such as: *multi-target binary classification model specification*, *multi-target multi-class classification model specification* (Demšar et al., 2006), and *multi-target regression model specification* (Kocev et al., 2009).

Finally, if we choose for example as a generalization language, the language of decision trees, we can define instances of all above mentioned predictive model specifications for the case of decision trees. All instances are presented in Figure 6.14. For example, a `multi-target regression tree specification` is an instance of the class *multi-target regression model specification* when we specify the language of decision trees as a generalization language.

6.3.3.3 Generalization as an implementation and generalization execution

In OntoDM-core, we define a *generalization* as a *realizable entity* that IS-CONCRETIZATION-OF a *generalization specification* and IS-SPECIFIED-OUTPUT-OF a *data mining algorithm execution*. It is represented by a *generalization representation* and IS-REALIZED-BY a *generalization execution* process. (see Figure 6.12). From the definition, we can see that generalizations have a dual nature. They can be treated as data structures and as such represented, stored and manipulated. On the other hand, they act as functions and are executed as they take input data examples and give as output the result of applying the function on a data example.

The dual nature of generalizations in OntoDM-core is represented with two classes that belong to two different description levels: *generalization representation*, which is a subclass of information content entity and belongs to the specification level, and *generalization execution*, which is a subclass of planned process and belongs to the application level.

A *generalization representation* IS-A IAO *information content entity* that IS-ABOUT a generalization. It represents a formalized description of the generalization, for instance in the form of a formula or text. For example, the output of a decision tree algorithm execution in any data mining software usually includes a textual representation of the generated decision tree.

A *generalization execution* IS-A OBI *planned process* that HAS-SPECIFIED-INPUT a *dataset* and HAS-SPECIFIED-OUTPUT another *dataset*. The output dataset is a result of applying the *generalization* to the examples from the input dataset. For example, in the case of a *predictive model*, the *predictive model execution* HAS-SPECIFIED-INPUT a *dataset* and gives as output a *predicted dataset*.

6.3.3.4 Example: representation of multi-target regression tree

In this section we present how we present an example of representation of a generalization instance in OntoDM-core. In Figure 6.15, we present the representation for multi-target regression tree in all aspects, including specification of the entity, implementation and application. A `multi-target regression tree specification` is an instance of a *multi-target regression model specification*. The specification is defined by the data specifications of the descriptive part and the output part of the data with instances denoting tuple of primitives and tuple of reals respectively. Next, the specification includes the language of decision trees as an instance of generalization language.

A `multi-target regression tree specification` IS-CONCRETISIZED-AS `multi-target regression tree ID0001`, which is an instance of *predictive model* class in the implementation description level. This instance IS-SPECIFIED-OUTPUT-OF a `multi-target regression tree algorithm execution ID0001`. Furthermore, `multi-target regression tree representation ID0001` is an instance that is about the regression tree and is used to contain its representation for the purpose of storage. Finally, `multi-target regression tree execution ID0001` is an instance of planned process and is used to represent the process of execution of a multi-target regression tree on a dataset and obtaining as product another dataset on which the functional part of the model has been applied to.

6.3.4 Data mining algorithm

A *data mining algorithm* is an algorithm (implemented in a computer program), designed to solve a data mining task. It takes as input a dataset of examples of a given datatype and produces as output a generalization (from a given type) on the given datatype. A specific data mining algorithm can typically handle examples of a limited set of datatypes: For example, a rule learning algorithm might handle only tuples of Boolean attributes and a boolean class.

Just as we have types of data, types of generalizations and data mining tasks, we have types of data mining algorithms. The latter are directly related to the input and output of the algorithm, but can depend also on the specifics of the algorithm, such as the basic components of the algorithm (e.g., heuristic function, search method). For example, for the decision tree building algorithms, we can have two sub-classes corresponding to top-down induction and beam-search.

A very desirable property of a data mining framework is to treat the mining of different types of structured data in a uniform fashion. In this context, data mining algorithms should be able to handle as broad classes of datatypes at the input as possible. We will refer to algorithms that can handle arbitrary types of structured data at the input as generic. Generic

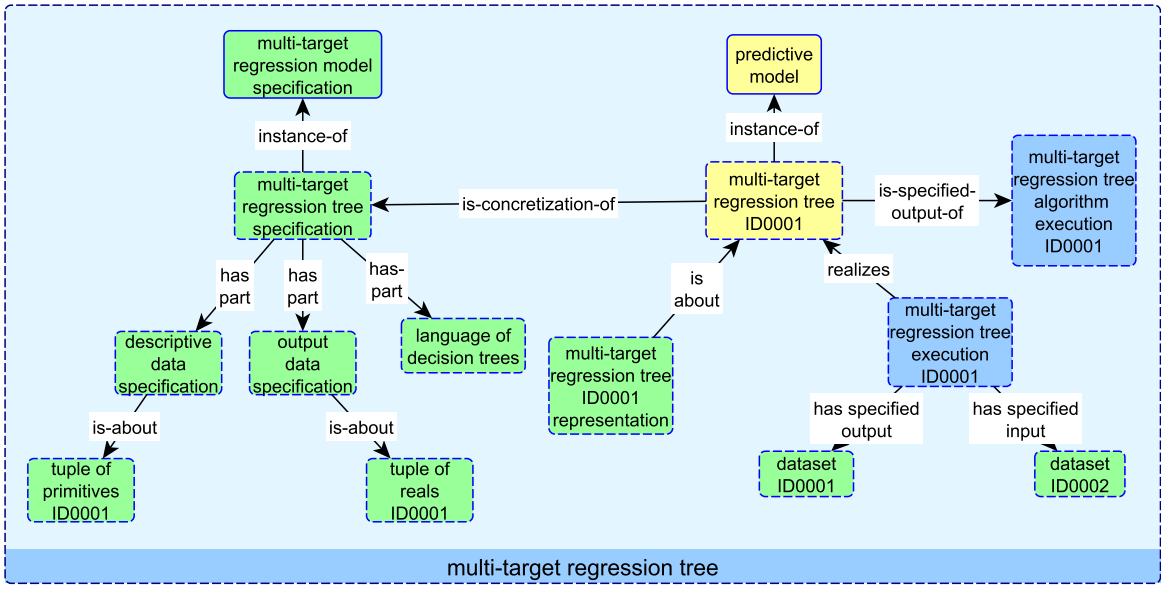


Figure 6.15: Example of representation of multi-target regression tree in OntoDM-core.

data mining algorithms would typically have as parameters some of their components, e.g., a heuristic function in decision tree induction or a distance in distance-based clustering.

In the context of modeling data mining algorithms in the OntoDM ontological framework, we define data mining algorithm is an algorithm, designed to solve a data mining task (See Figure 6.16). We consider three aspects of this entity: a DM algorithm (as a specification), a DM algorithm implementation, and a DM algorithm execution. Each aspect gives rise to a separate entity: Due to the different nature of these entities, each has a different parent class and is modeled in a different module of the ontology.

In the remainder of this subsection, we review all three aspects and show how they are interconnected. First, we analyze the data mining algorithm as a specification and present a taxonomy of data mining algorithms (Section 6.3.4.1). Next, we focus on two extensions of the taxonomy of algorithms in order to represent more specifically algorithms for predictive modeling (Section 6.3.4.2) and algorithms for hierarchical classification (Section 6.3.4.3). Furthermore, we look at data mining implementation and the process of data mining algorithm execution (Section 6.3.4.4). Finally, at the end of this section we present a generalized structure for representing algorithms in computer science (Section 6.3.4.5).

6.3.4.1 Data mining algorithm as a specification and taxonomy of data mining algorithms

A *data mining algorithm* (as a specification) IS-A IAO *plan specification* that HAS-PART a *data mining task*, an IAO *action specification*, a *generalization specification*, and a IAO *document* (see Fig. 6.16). The *data mining task* defines the objective that the realized plan should fulfill at the end giving as output a generalization, while the *action specification* (as defined in IAO) "is a directive information entity that describes the action the bearer will take", in other words the actions of the data mining algorithm realized in the process of execution. The *generalization specification* denotes the type of generalization produced by executing the algorithm. Finally, having a *document* class as a part allows us the opportunity to connect to the annotations of documents (journal articles, workshop articles, technical reports) that publish the knowledge about the algorithm.

In analogy with the taxonomy of datasets, data mining tasks and generalizations, in OntoDM-core we also construct a taxonomy of data mining algorithms (see Figure 6.17). As criteria, we used the data mining task and the generalization produced as the output of

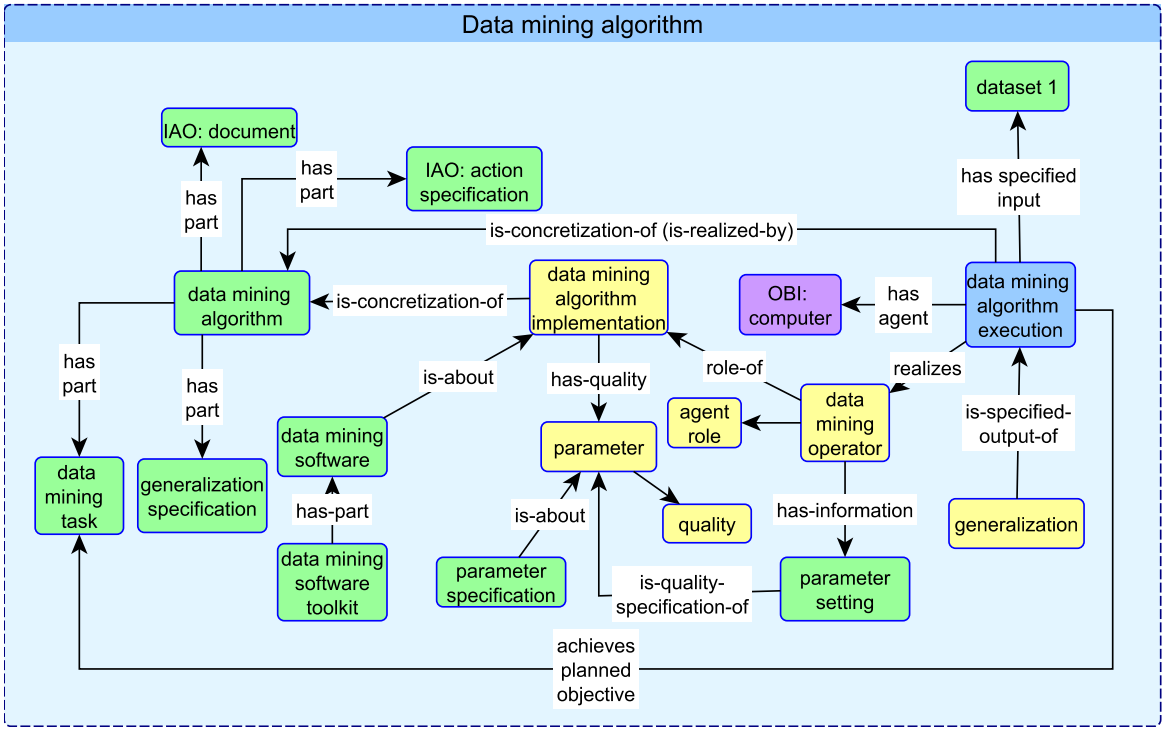


Figure 6.16: Representation of a data mining algorithm in OntoDM.

the execution of the algorithm. Since both data mining task and generalization specification include specifications of the datatypes, this taxonomy of data mining algorithms is general and can cover (represent) algorithms that operate on arbitrary types of data.

At the first level of OntoDM-core taxonomy of data mining algorithm we distinguish between *single generalization algorithms* and *ensemble algorithms*. A *single generalization algorithm* specifies a data mining algorithm that solves a data mining task and as result produces a single generalization. An *ensemble algorithm* specifies a data mining algorithm that solves a data mining task and as a result produces an ensemble of generalizations. In addition, an *ensemble algorithm* has as part *single generalization algorithms*.

At the second level, the algorithm structure is extended to cover specific algorithms task wise. For example, *single generalization algorithm* is sub-classed with *pattern-discovery algorithm*, *clustering algorithm*, *predictive modeling algorithm*, and *probability distribution estimation algorithm* (see Figure 6.17). In the language of OntoDM-core, these are defined as follows:

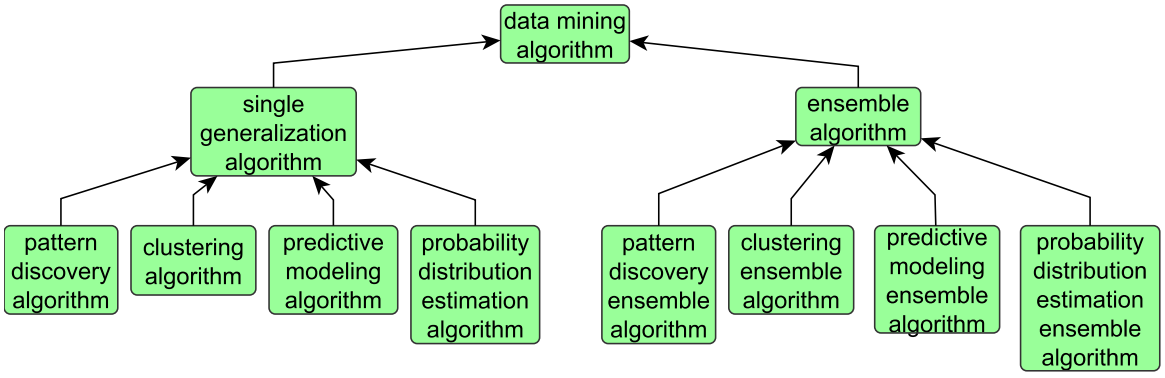


Figure 6.17: Taxonomy of data mining algorithms in OntoDM-core.

- A *pattern discovery algorithm* is defined as a single generalization algorithm that solves a pattern discovery task and produces as a result a set of patterns;
- A *clustering algorithm* is defined as a single generalization algorithm that solves a clustering task and produces as a result a clustering;
- A *predictive modeling algorithm* is defined as a single generalization algorithm that solves a predictive modeling task and produces as a result a predictive model; and
- A *probability distribution estimation algorithm* is defined as a single generalization algorithm that a solves probability distribution estimation task and produces as a result a probability distribution.

In analogy, for the case of ensemble algorithms we define four different sub-classes of ensemble algorithms (see Figure 6.17). In the language of OntoDM-core, these are defined as follows:

- A *pattern discovery ensemble algorithm* is defined as an ensemble algorithm that solves a pattern discovery task and produces as a result an ensemble of set of patterns;
- A *clustering ensemble algorithm* is defined as an ensemble algorithm that solves a clustering task and produces as a result a clustering ensemble;
- A *predictive modeling ensemble algorithm* is defined as an ensemble algorithm that solves a predictive modeling task and produces as a result an predictive models ensemble; and
- A *probability distribution estimation ensemble algorithm* is defined as an ensemble algorithm that a solves probability distribution estimation task and produces as a result a probability distribution ensemble.

6.3.4.2 Taxonomy of predictive modeling algorithms

In sections 6.3.2.2 and 6.3.3.2, we presented our taxonomy of predictive modeling and taxonomy of predictive models respectively. In line with those two proposals we construct a taxonomy of predictive modeling algorithms (see Figure 6.18). The taxonomy of predictive modeling algorithms is an extension of the more general taxonomy of data mining algorithms and in its structure it follows the structure of taxonomy of PM tasks and PM, and is due to the structure of the *data mining algorithm* class having as parts *data mining task* and *generalization specification*.

By using the data mining task as a criterion, at the first level we distinguish between the *predictive modeling algorithm for primitive output prediction* and the *predictive modeling algorithm for structured output prediction*. In the first case, the data mining task the algorithm is solving is *primitive output prediction task*, and in the second case, it *structured output prediction task*.

Predictive modeling algorithm for primitive output prediction. The *predictive modeling algorithm for primitive output specification* can be feature-based or structure-based, depending on the data mining task it is solving. *Feature-based predictive modeling algorithm for primitive output prediction* specifies an algorithm that solves the *feature-based primitive output prediction task*. This is the most exploited type of algorithms in traditional single-table data mining, described in all major data mining textbooks (e.g., Hand et al. (2001)).

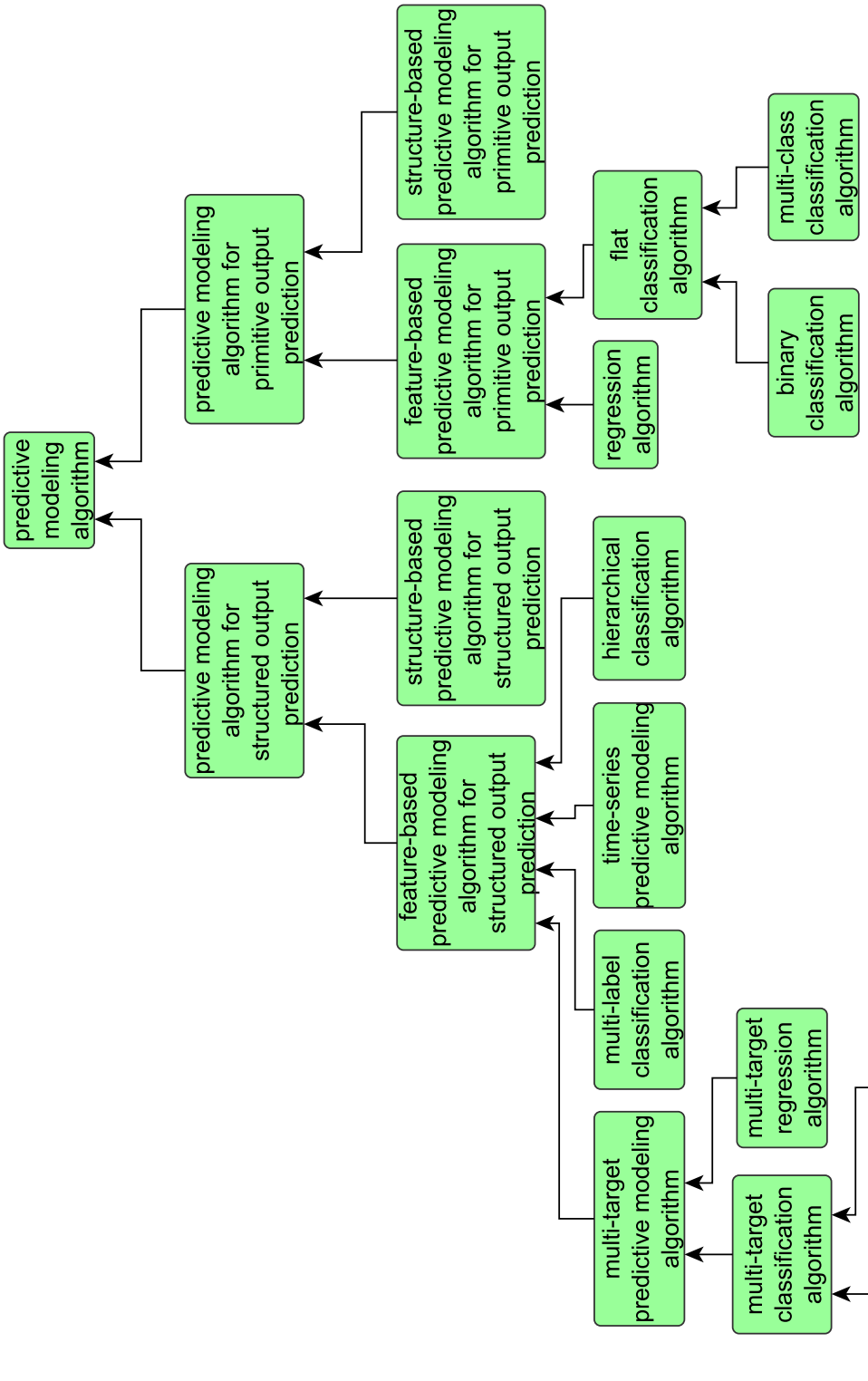


Figure 6.18: Taxonomy of predictive modeling algorithms in OntoDM-core.

If we specify the data mining task in in more detail, we can represent other classes of algorithms such as the *binary classification algorithm*, the *multi-class classification algorithm* and *regression algorithm*; where the data mining task is *binary classification*, *multi-class classification*, and *regression task*, respectively. *Structure-based predictive algorithm for primitive output prediction* specifies a class of algorithms that solve the *structure-based primitive output prediction task*.

Predictive modeling algorithm for structured output prediction. In similar way, *predictive modeling for structured output prediction* can be feature-based or structure-based. *Feature-based predictive modeling algorithm for structured output prediction* specify a class of algorithms that solve the *feature-based structured output prediction task*. *Structure-based predictive modeling algorithm for structured output prediction* specify class of algorithms that solve the *structure-based structure output prediction task*.

If we focus just on feature-based algorithms and further specify the data mining task at hand, we can represent a variety of structured output predictive modeling algorithms. For example, we can represent the following classes of algorithms: *multi-target predictive modeling algorithm* (Caruana, 1997) (which solves the *multi-target prediction task*), *multi-label classification algorithm* (Tsoumakas and Katakis, 2007) (which solves the *multi-label classification task*), *time-series predictive modeling algorithm* (Slavkov et al., 2010) (which solves the *time-series prediction task*) and *hierarchical classification algorithm* (Silla and Freitas, 2011) (which solves the *hierarchical classification task*). Furthermore, a *multi-target predictive modeling algorithm* can be further specified into algorithm classes such as: *multi-target binary classification algorithm*, *multi-target multi-class classification algorithm* (Demšar et al., 2006), and *multi-target regression algorithm* (Kocev et al., 2009).

On the lower levels, we can extend the taxonomy by taking into account the type of generalization as a criteria. For example, we can define a *multi-target regression tree algorithm* class as a sub-class of *multi-target regression algorithm* that has a specifics that it produces as output a *multi-target regression tree*.

6.3.4.3 Further extensions: Taxonomy of hierarchical classification algorithms

The taxonomy of predictive modeling algorithms, discussed in the previous section, can be further extended on the basis of other sub-domain criteria. In line with the proposal of a taxonomy of hierarchical classification tasks presented in Section 6.3.2.3, in this section, we present a proposal how we can further extend the taxonomy of hierarchical classification algorithms by following the taxonomy of tasks and using the criteria presented by Silla and Freitas (2011), in their review paper on hierarchical classification across different application domains.

In Figure 6.19, we present the taxonomy of hierarchical classification algorithms. At the first level of the taxonomy, we have a distinction between *tree-based hierarchical classification algorithms* and *DAG based hierarchical algorithm*. The first class of algorithms solve the *tree-based hierarchical classification task*, while the second class of algorithms solve the *DAG based hierarchical classification task*.

If we just focus on the DAG based algorithms (the structure of taxonomy for tree based algorithms is analogous), at the second level of the taxonomy we have *DAG based HC algorithm for single path prediction* and *DAG based HC algorithm for multiple path prediction*. Here the criteria for the division is information whether the algorithm solves the task that allows class labels that are associated with single or multiple paths in the class hierarchy. Furthermore, we have a division into algorithms that have mandatory and non-mandatory leaf node prediction.

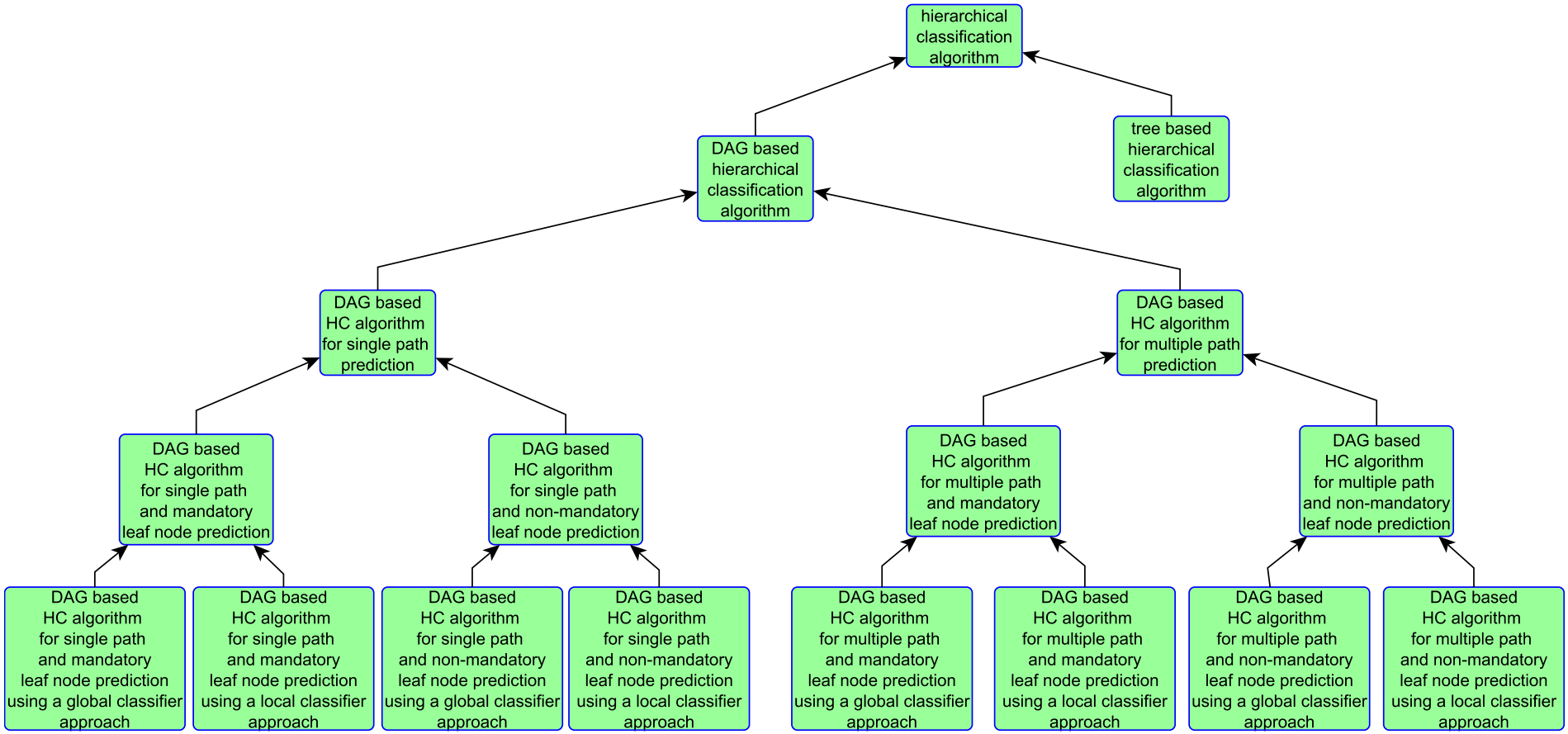


Figure 6.19: A part of the taxonomy of hierarchical classification algorithms in OntoDM-core. The figure contains only part of the taxonomy dealing with the DAG based algorithm. The structure of the tree based algorithm taxonomy is analogus.

For example, *DAG based HC algorithm for single path prediction* is further extended with the following classes of algorithms: *DAG based HC algorithm for single path and mandatory leaf node prediction* and *DAG based HC algorithm for single path and non-mandatory leaf node prediction*. Finally, at the last level of the taxonomy we distinguish between algorithms that use a global classifier approach and local classifier approach (see (Silla and Freitas, 2011) for more information about each approach). For example, *DAG based HC algorithm for single path and mandatory leaf node prediction* can use a global classifier aproach or a local classifier approach.

For example, in OntoDM-core we can represent the proposal of Vens et al. (2008) for an algorithm for learning decision trees for hierarchical multi-label classification. This algorithm is solving a *DAG based HC task for multiple paths and partial depth labeling* (see Section 6.3.2.3) and produces as output a hierarchical classification decision tree. The algorithm uses a global classifier approach, that is why it is an instance of the *DAG based HC algorithm for multiple path and non-mandatory leaf node prediction using a global classifier approach class*.

6.3.4.4 Data mining algorithm implementation, operator, software, and execution

In the previous sections, we discussed the data mining algorithm entity from the perspective of the specification description level and presented a way of establishing a taxonomy of data mining algorithms based on different criteria. In this section, we focus on the implementation and application description level in the context of data mining algorithms.

Data mining algorithm implementation. In OntoDM-core, a *data mining algorithm implementation* is defined as a sub-class of BFO *realizable entity*. Furthermore, a *data mining algorithm implementation* IS-CONCRETIZATION-OF a *data mining algorithm*, in the form of a runnable computer program. A *data mining algorithm implementation* HAS-QUALITY *parameters* (see Figure 6.16). The parameters of the algorithm affect the possible behavior of an algorithm when the algorithm implementation is used as an operator. For example, `wekaM5P` is an instance of a *data mining algorithm implementation* class, and it is a concretization of a `regression tree algorithm`. This instance has as quality parameters such as `wekaM5P-unpruned tree parameter`, `wekaM5P-unsmoothed predictions parameter`, `wekaM5P-minimum number of instances parameter`, and others (see Fig. 6.20). Finally, *parameter specification* is a specification entity about a parameter and specifies things such as name, description and others.

Data mining operator. In OntoDM-core, a *data mining operator* is defined as sub-class of the *role* class . In that context, a *data mining operator* IS-ROLE-OF a *data mining algorithm implementation* that IS-REALIZED-BY a *data mining algorithm execution*. For example, `wekaM5P operator ID001` is an instance of the *data mining operator* class (see Figure 6.20). It is a role of the `wekaM5P` implementation in the context of its execution (realization) and by the `wekaM5P algorithm execution ID001` process instance.

Finally, the *data mining operator* HAS-INFORMATION about the specific *parameter setting* of the algorithm, in the context of realization of the operator in the process of execution. The *parameter setting* IS-A *data item* which IS-QUALITY-SPECIFICATION-OF a *parameter*. For example, `wekaM5P unpruned tree parameter setting ID001` is an instance of parameter setting, and represents a quality specification of a `wekaM5P unpruned tree parameter` that is provided as information to the operator in order for it to be executed in a process (see Figure 6.20).

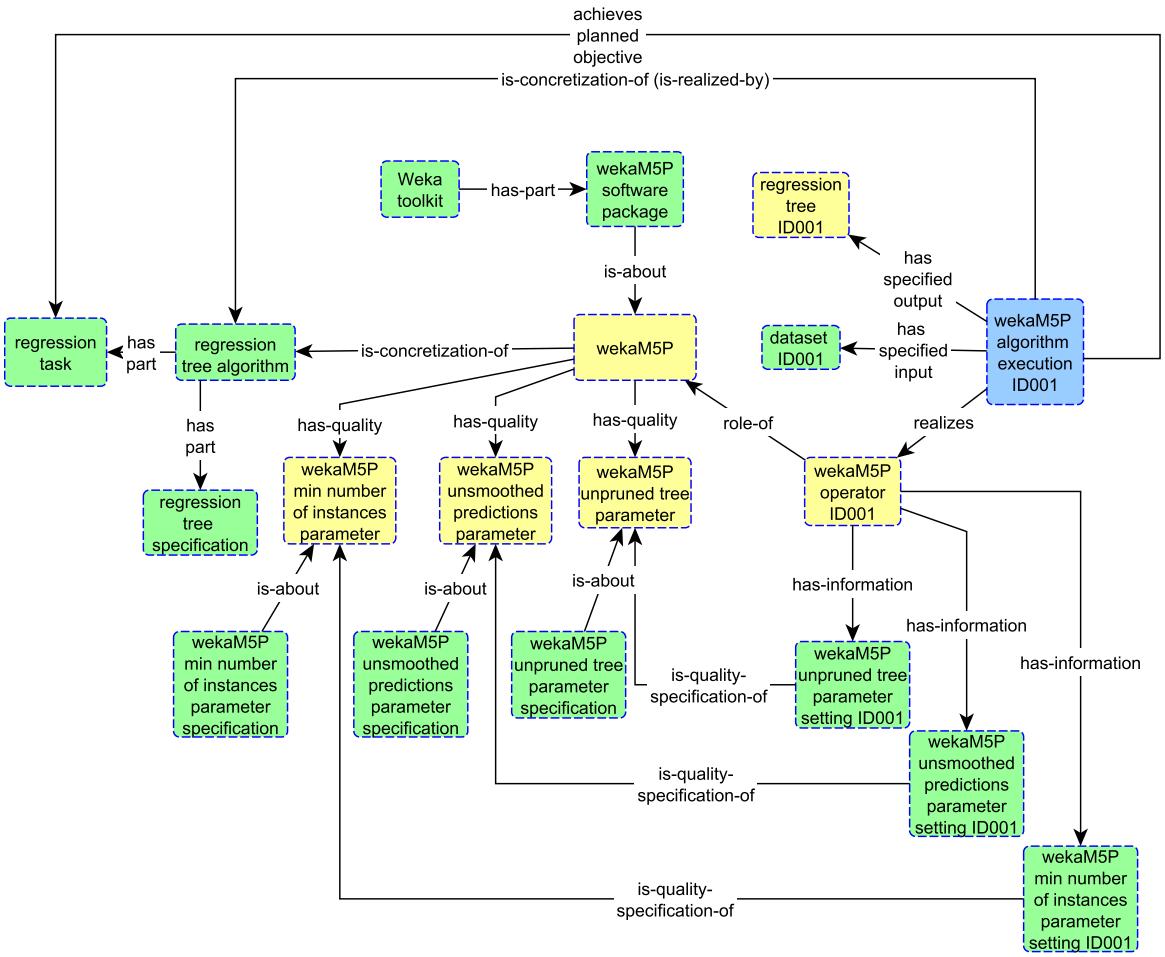


Figure 6.20: Representation of of wekaM5P instance of a data mining algorithm implementation.

Data mining software. In OntoDM-core, a *data mining software* is defined as a sub-class of IAO *directive information entity*. It represents a specification of a *data mining algorithm implementation* (see Figure 6.16). It has as parts all the meta-information entities about the software implementation such as: *source code*, *software version specification*, *programming language*, *software compiler specification*, *software manufacturer*, the *data mining software toolkit* it belongs to, etc. Finally, a *data mining software toolkit* is an specification entity that contains as parts *data mining software* entities. For example, `weka toolkit` is an instance of *data mining software toolkit* class and has as part `wekaM5P software package`, which is an instance of *data mining software* (see Figure 6.20).

Data mining algorithm execution In OntoDM-core, we define a *data mining algorithm execution* as a sub-class of OBI *planned process* class. A *data mining algorithm execution* REALIZES a *data mining operator*, HAS-SPECIFIED-INPUT a *dataset*, HAS-SPECIFIED-OUTPUT a *generalization*, HAS-AGENT a *computer*, and finally ACHIEVES-PLANNED-OBJECTIVE a *data mining task* (see Figure 6.16). For example, `weka M5P algorithm execution ID001` REALIZES a `wekaM5P operator ID001`, it HAS-SPECIFIED-INPUT `dataset ID001`, it HAS-SPECIFIED-OUTPUT `regresion tree ID001`, and ACHIEVES-PLANNED-OBJECTIVE `regression task` (see Figure 6.20). Finally, having a *computer* as an agent of the data mining algorithm execution process, allows us to further extend the information about the execution of an algorithm with the meta-information about the computer that is active in the execution process. This includes entities such as: computer name, type of processor, information about processing time, and other information.

To conclude, the structure presented in Figure 6.16 is adequate for the representation of the execution of both simple and complex algorithms (the latter using other algorithms in their execution), such as ensemble learning algorithms. The major design notion here is that a *data mining algorithm execution* process in the case of ensemble learning algorithms has as parts sub-processes which are executions of the simple (base level) algorithms.

6.3.4.5 Proposal for a generic structure for representing algorithms in computer science

In the previous sections, we presented the OntoDM-core classes that are used to represent data mining algorithms at the three description levels: specification, implementation and application. The mechanism for representing data mining algorithms is very general and can be easily modified to represent arbitrary algorithms in computer science. These representations can be further reused and extended by other ontologies. In Figure 6.21, we present the general structure for representing algorithms, which is independent of the nature and the origin of the algorithms.

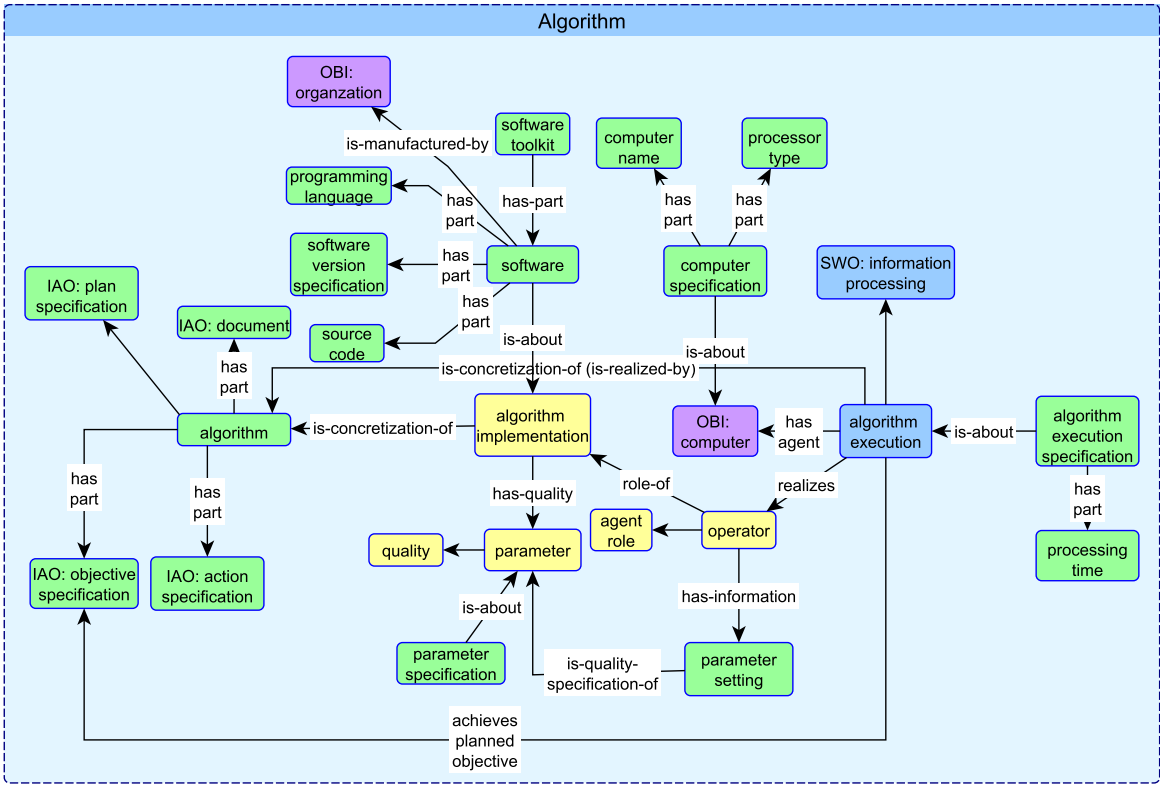


Figure 6.21: General structure for representation of algorithms for computer science.

As we can see from Figure 6.21, an *algorithm* is an IAO *plan specification* and contains as parts IAO *objective specification*, IAO *action specification*, and IAO *document* classes. An *algorithm implementation* is a OBI *plan*, which is concretization of an algorithm, and has quality *parameters*, that are specified by a *parameter specification*. *Software* entity is a specification of an algorithm implementation and has parts *source code*, *software version specification*, *programming language specification*, and IS-MANUFACTURED-BY some *organization*. On the other hand, *operator* IS-ROLE-OF an *algorithm implementation* which IS-REALIZED-BY the process of *algorithm execution* by providing information about *parameter setting* of each parameter defined by the implementation. Finally, *algorithm execution* is a SWO *information processing process* that REALIZES an *operator* and depending on the algorithm at hand has defined inputs ans outputs and has agent a *computer*. In addition, we

define computer specification as an entity about a computer that characterizes computer's specifics, and we define algorithm execution specification that is about a specific algorithm execution and can contain information such as processing time.

6.3.5 Constraints and constraint-based data mining task

Constraints play a central role in data mining and constraint-based data mining is now a recognized research topic (Bayardo, 2002). The use of constrains enables more efficient induction of generalizations and focuses the search for generalizations that are likely to be of interest to the end user. Mannila and Toivonen (1997) state that a general statement of the problem of data mining involves the specification of a language of generalization and a set of constraints that a generalization needs to satisfy. In constraint-based data mining constraints are propositions or statements about generalizations. In this section, we first present the OntoDM-core taxonomy of constraints (Section 6.3.5.1). Next, we focus on constraint-based data mining tasks and a taxonomy of constraint-based data mining tasks (Section 6.3.5.2).

6.3.5.1 Taxonomy of constraints

In OntoDM-core, we define a *constraint specification* as a sub-class of OBI *data representational model*. Many types of constrains are used in constraint-based data mining and can be classified along several orthogonal dimensions. First, we distinguish between primitive and composite constraints. Next, we distinguish between language and evaluation constraints. Finally, we distinguish among hard (Boolean) constraints, soft constraints and optimization constraints. In the rest of this section, we briefly overview the different dimensions and present the OntoDM-core taxonomy of constrains.

Constraints in constraint-based data mining represent propositions on generalizations. Propositions can be atomic or non-atomic. Atomic propositions are not decomposable into smaller propositions, while the non-atomic propositions are decomposable. The constrains that are based on atomic propositions are called primitive constraints. Consequently, the constraints that are based on non-atomic propositions are called composite constraints. The primitive constrains can be combined using boolean operations and form composite constraints.

The constraints in constraint-based data mining refer to either the form (syntax) of a generalization or the its validity (semantics) with respect to the data. If the constrains refer to the syntax of the generalization then they are called language constraints, and if they refer to the semantics they are called evaluation constraints. Language constrains concern the representation of a generalization and only refer to its form. A commonly used type of language constraints is the subsumption constraint. Other type of language constrains can include a cost function on patterns and models. The cost functions, in this case, represent mappings from the representation of the generalization to a non-negative real. Evaluation constraints, on the other hand, concern the semantics of a generalization when applied to a dataset. They usually include evaluation functions, where the evaluation functions measure the validity of a generalization on a given dataset. The evaluation functions take as input a generalization and return a real value as output. The above mentioned types of constraints are definitions of primitive language and evaluation constraints. They can be used to form composite language constraints, composite evaluation constraints, and composite constraints that have a mix of primitive language and evaluation constraints.

In addition, we distinguish between hard constraints, soft constrains, and optimization constraints. Hard constraints represent boolean functions on generalizations. In this case the constraint can be either satisfied or not satisfied. Soft constraints, on the other hand, do not dismiss a generalization that violates a constraint but rather the generalization is penalized for violating a constraint. Optimization constrains allow us to ask for a set of

generalization (of a fixed size) that have some extreme value (minimum or maximum) for a given cost or evaluation function.

In Figure 6.22, we present the OntoDM-core taxonomy of constraints. At the first level we have the *primitive constraint* and *complex constraint*. *Complex constraints* have as parts *primitive constraints*nd a *combination function specification* that specifies the way how the primitive constraints are combined to form a complex constraint. At the second level, if we focus in the primitive constraints, we have *primitive language constraints* and *primitive evaluation constraints*. Furthermore, *primitive language constraints* class is further extended with *primitive subsumption based language constraint* and *primitive language cost-function constraint* class. At the last level the *primitive language cost-function constraint* is extended with three sub-classes that include: *primitive hard language cost-function constraint*, *primitive soft language cost-function constraint*, and *primitive optimization language cost-function constraint*. In similar way, we define sub-classes of *primitive evaluation constraint* class.

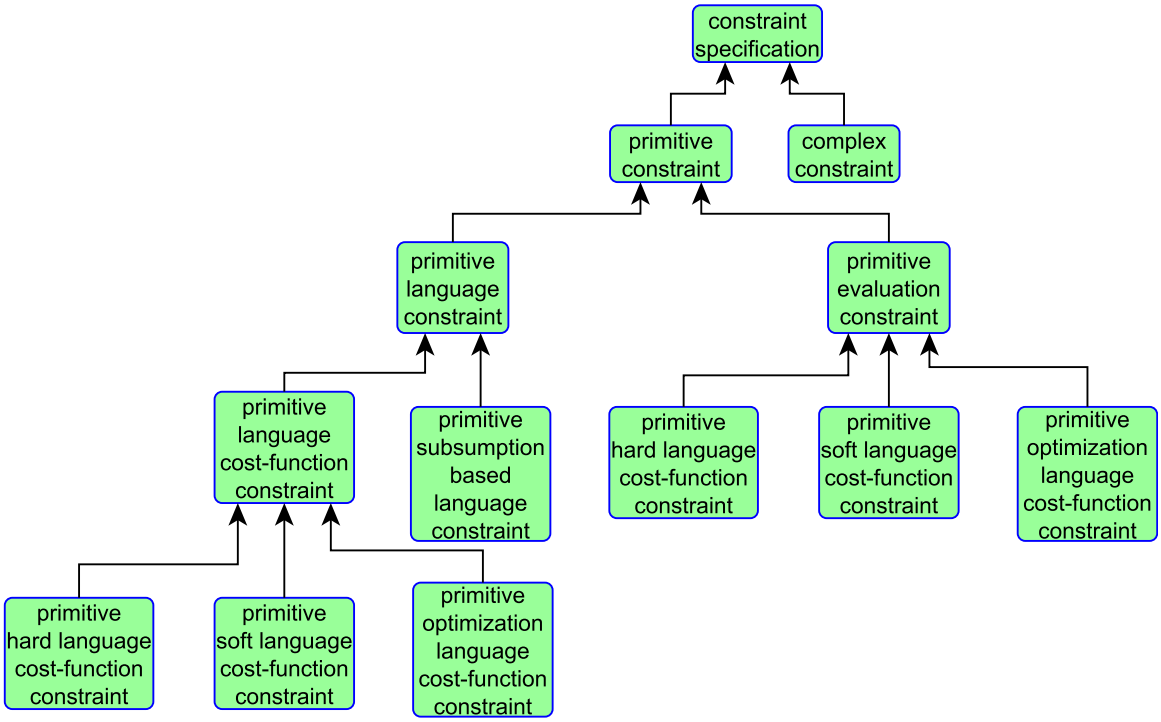


Figure 6.22: Taxonomy of constraints.

6.3.5.2 Constraint-based data mining task and taxonomy of constraint-based data mining tasks

The task of constraint-based data mining (CBDM) is to find a set of generalizations that satisfy a set of constraints, given a dataset that consists of examples of a specific datatype, a data mining task, a generalization specification and a specifications of the set of constraints. In this context, we can see that a constraint-based data mining task covers the whole domain of data mining, and the traditional formulation of data mining is a special case of this formulation. The major difference between the traditional and the constraint-based data mining paradigms is that in the traditional paradigm we usually have one evaluation metrics and the algorithms give as output only one solution. In the remaining of this section, we present the representation of CDBM task and algorithm in OntoDM-core.

In the OntoDM-core ontology module, we represent a *CBDM task* as a sub-class of IAO *objective specification* class (see Figure 6.23). It has as parts a *data mining task* and set of *constraint specifications*. We further define a *CBDM algorithm* as an algorithm that

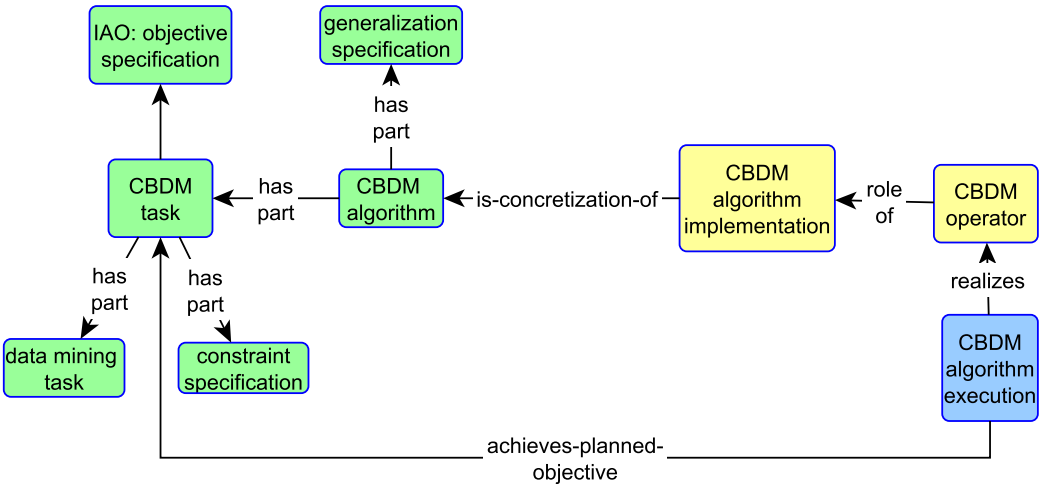


Figure 6.23: Constraint-based data mining task and algorithm in OntoDM-core.

solves a CBDM task. Furthermore, we represent a *CBDM algorithm implementation* which is a concretization of a *CBDM algorithm*. Finally, *CBDM operator* is a role of a *CDBM algorithm implementation* and is realized in the *CBDM algorithm execution* process, that achieves planned objective a *CDBM task*.

From Figure 6.23, we can see that the definition of the *CBDM task* class contains as part *data mining task*. This allows us to form a taxonomy of CBDM tasks. In Figure 6.24, we present part of the taxonomy of CBDM tasks. At the first level of the taxonomy, we have the basic CBDM task classes that are aligned with the foundational data mining task. These include *constraint-based pattern discovery*, *constraint-based clustering*, *constraint-based predictive modeling*, and *constraint-based probability distribution estimation*. At the second level, if we focus on constraint-based pattern discovery we can extend the taxonomy with the following classes depending on the descriptive data specification. This includes *constraint-based itemset mining*, *constraint-based sequence mining*, *constraint-based tree mining*, and *constraint-based graph mining*. Examples of constraint-based itemset mining task includes tasks such as *frequent itemset mining*, *closed itemset mining*, *discriminative itemset mining*, and *cost-based itemset mining*.

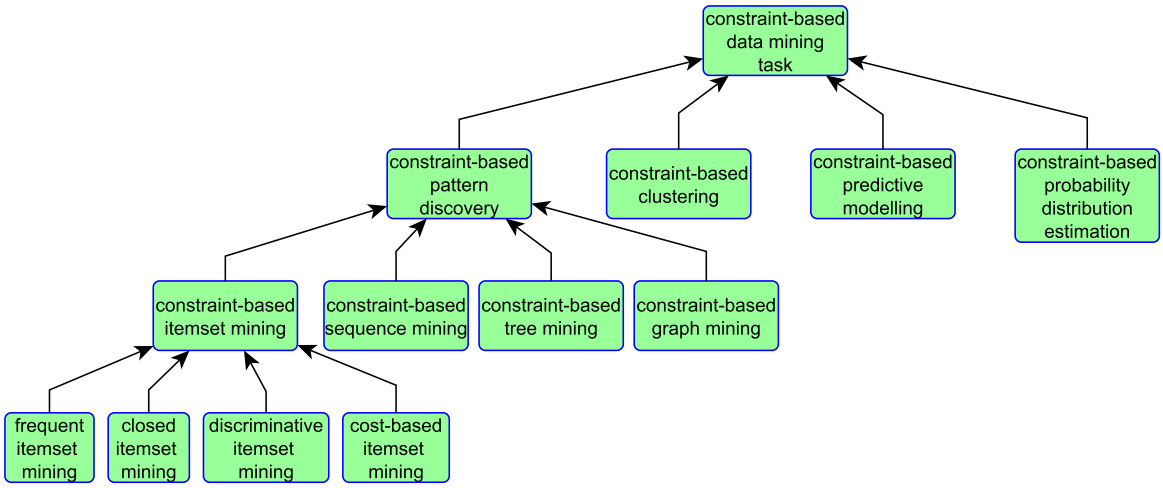


Figure 6.24: Part of the taxonomy of constraint-based data mining tasks.

6.3.6 Data mining scenario

A scenario is usually defined in the dictionary as "a postulated sequence or development of events"[1]. Therefore, a data mining scenario deals with the logical sequence of processes to infer some type of generalization from data. In the OntoDM-core module, we represent a data mining scenario with the different aspects, reflected in entities at different description levels in the ontology: data mining scenario (as a specification), data mining workflow (as an implementation), and data mining workflow execution.

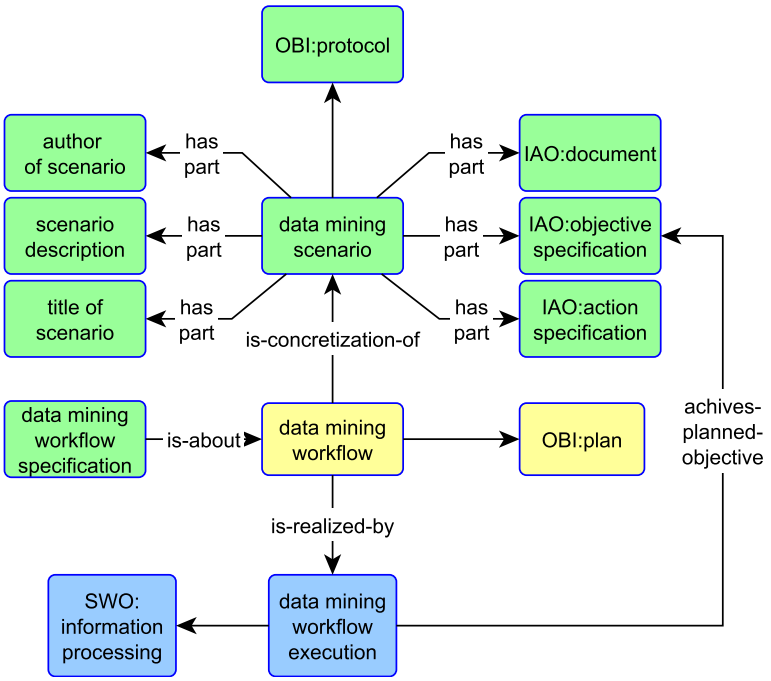


Figure 6.25: Scenarios and workflows in OntoDM.

In OntoDM-core (see Figure 6.25), a *data mining scenario* IS-AN extension of the OBI *protocol* class, and represents an information entity. It includes as parts other information entities such as: *title of scenario*, *scenario description*, *author of scenario*, and IAO *document*. Additionally, from the protocol class it inherits as parts IAO *objective specification* and *action specification*. All these classes are used to describe a scenario. Furthermore, a *data mining workflow*, in OntoDM, IS-CONCRETIZATION-OF of a data mining scenario, and it is an extension of the OBI *plan* entity. Finally, a data mining workflow IS-REALIZED-BY *data mining workflow execution*, which extends the SWO class *information processing*. The ordering of sub-processes in a workflow execution is determined by using the PRECEDED-BY relation.

[1] `http://oxforddictionaries.com/definition/scenario`

7 OntoDM-KDD: Ontology Module for Data Mining Investigations

The term knowledge discovery in databases, or KDD for short, refers to the broad process of finding knowledge in data. The term data mining refers to the sub-process of the KDD process that involves the application of algorithms for extraction of knowledge from data in form of patterns (generalizations). The KDD process is interactive and iterative. It involves numerous sub-processes, such as for example: developing an understanding of the application domain, the relevant prior knowledge, and end user's goals; collecting data and creating a target dataset; data cleaning and pre-processing; data reduction and projection; choice of data mining task; choice of data mining algorithm; modeling or data mining; deployment of mined knowledge; and incorporating and use of the mined knowledge.

In this chapter, we present the OntoDM-KDD ontology module. First, we present the goal, scope and competencies of OntoDM-KDD (Section 7.1). Next, we discuss the implementation and availability of the OntoDM-KDD ontology module (Section 7.2). Finally, we describe in more detail the structure of the ontology module (Section 7.3).

7.1 Goal, scope and competencies

OntoDM-KDD is an ontology module for representing data mining investigations. Its goal is to allow the representation of knowledge discovery processes and be general enough to represent the data mining investigations.

All of the sub-processes in the KDD process are essential to ensure that useful knowledge is derived from the data. In the domain of data mining and knowledge discovery, there are proposals for standardizing the process of knowledge discovery in the context of representing and performing data mining investigations. One of the most prominent proposals is the CRISP-DM methodology (Chapman et al., 1999). CRISP-DM stands for Cross Industry Standard Process for Data Mining. It is a process model that describes data mining investigations performed in practical applications. The CRISP-DM process model is based on commonly used approaches that expert data miners use to tackle and solve the practical problems in the domain of data mining.

Chapman et al. (1999) describe the CRISP-DM methodology in terms of a hierarchical process model, consisting of a set of tasks described at four levels of abstraction: phase, generic task, specialized task, and process instance. At the top level, the process is organized into phases, where each phase consists of several second-level generic tasks. The third level, the level of specialized tasks, describes how the generic tasks should be carried out in specific situations. The fourth level, the level of process instances, is a description of actions, decisions and results of an actual data mining investigation.

In general, life cycle of a data mining investigation consists of six phases (see Fig. 7.1). The sequence of phases is not rigid. Iterating back and forth between phases is required and determined by the outcome of each phase separately. The phases include : the application (or business) understanding phase, data understanding phase, data preparation phase, modeling phase, evaluation phase, and deployment phase.

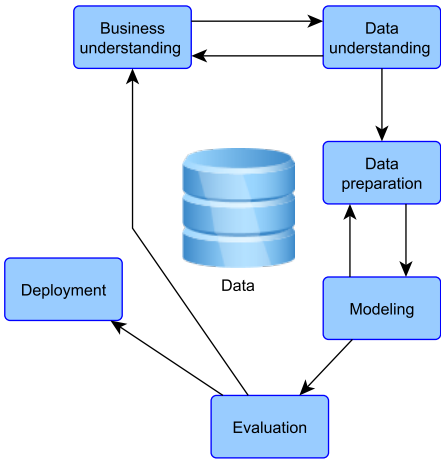


Figure 7.1: The CRISP-DM process model.

Based on the main goal of the OntoDM-KDD module, we established a list of competency questions our ontology module is designed to answer. The most important competency questions are listed in Table 7.1. Judging from the list of questions, the ontology will include information about the basic information entities for representing data mining investigations, such as action specifications, objective specifications, plan specifications, reports, and textual entities. Furthermore, the module will contain processual entities that compose the knowledge discovery process, such as acquisition, documenting, identification, information processing, interpreting data, planning, selection, and validation.

Table 7.1: OntoDM-KDD competency questions.

Q	**Competency Question**
1	What is the investigation title/description of a DM investigation X?
2	What is the set of publications about the investigation for a DM investigation X?
3	What is the DM investigation that is reported by a publication X?
4	What is output of a KD phase X for a DM investigation Y?
5	What is the output of a sub-process X of KD phase Y for a DM investigation Z?
6	What is the set of actions realized by the KD phase X for a DM investigation Y?
7	What is the set of DM investigations that realize an action X in a KD phase Y?
8	What is the process that precedes process X in KD phase Y for a DM investigation Z?

7.2 Implementation

In order to represent data mining investigations, we model the CRISP-DM process in the OntoDM-KDD ontology module by reusing and extending upper level classes from the OBI, the IAO and the SWO ontology. This is compliant with the design principles discussed in Chapter 4. In the OntoDM-KDD ontology module, we distinguish two levels of representa-

tion (see Figure 7.2 and 7.3). The first level is the specification level (Section 7.2.1), that deals with information entities needed to describe and represent the data mining investigations. The second level is the application level (Section 7.2.2) that deals with processual entities in order to represent processes that occur in a data mining investigation.

7.2.1 Specification level

The specification level (see Figure 7.2) consists of classes that are extensions of IAO class *information content entity*. At the top level, it includes classes such as *data item*, *directive information entity*, *document*, and *textual entity*. The *directive information entity* class is further extended with *action specification*, *data format specification*, *objective specification*, and *plan specification* classes (see Section 4.2.2.1). These top level classes are further subclassed in order to provide a representational mechanism to support the representation of data mining investigations.

For example, the *action specification* class is further extended with a taxonomy of KDD specific actions realized in the KDD processes. At the first level (see Figure 7.2), we have the more general action specification classes such as *analyze action*, *assess action*, *check action*, *compare action*, *describe action*, *document action*, *interpret action*, *plan action*, *process information action*, *revise action*, and *specify action*. At the second level, the general action specification classes are extended with KDD specific classes of actions. For example, the *assess action* specification is extended with the following sub-classes: *assess attribute importance*, *assess data sources*, *assess modeling assumptions*, and others.

In addition, in order to support representation of data mining investigations, from the OBI ontology we reuse *study design* and *protocol* classes (see Section 4.2.2.2), that are extension od *plan specification*. We further extend these classes with *KD phase design*, containing design specifications of knowledge discovery phases defined in the CRISP-DM methodology, and *KD protocol* containing knowledge design protocols that are defined for each of the KD phases.

7.2.2 Application level

The application level consists of classes that are extensions of the OBI class *planned process*. We extend the *planned process* class with the *KDD phase* class and its sub-classes, in order to represent the different phases in a data mining investigation.

In Figure 7.3, we present the OntoDM-KDD processual taxonomy. At the top level, it includes classes for representing different types of general processes. These include: *validation*, *planning*, *interpreting data*, *information processing*, *selection*, *identification*, *documenting*, *acquisition*, *study design execution*, and *investigation*. The *study design execution* class is further extended with *KD phase*, that represents the process of execution of KD phases defined in the CRISP-DM methodology and includes the following sub-classes: *application understanding*, *data understanding*, *data preparation*, *modeling*, *DM process evaluation*, and *deployment*.

In addition, at the second level of the process taxonomy we have the KD specific processes, that represent realizations of the KD protocols defined for each KD phase. For example, the *validation* process has as sub-classes *results evaluation*, *project review*, *model assessment*, *data quality verification*, *business situation assessment*, and *DM process review*.

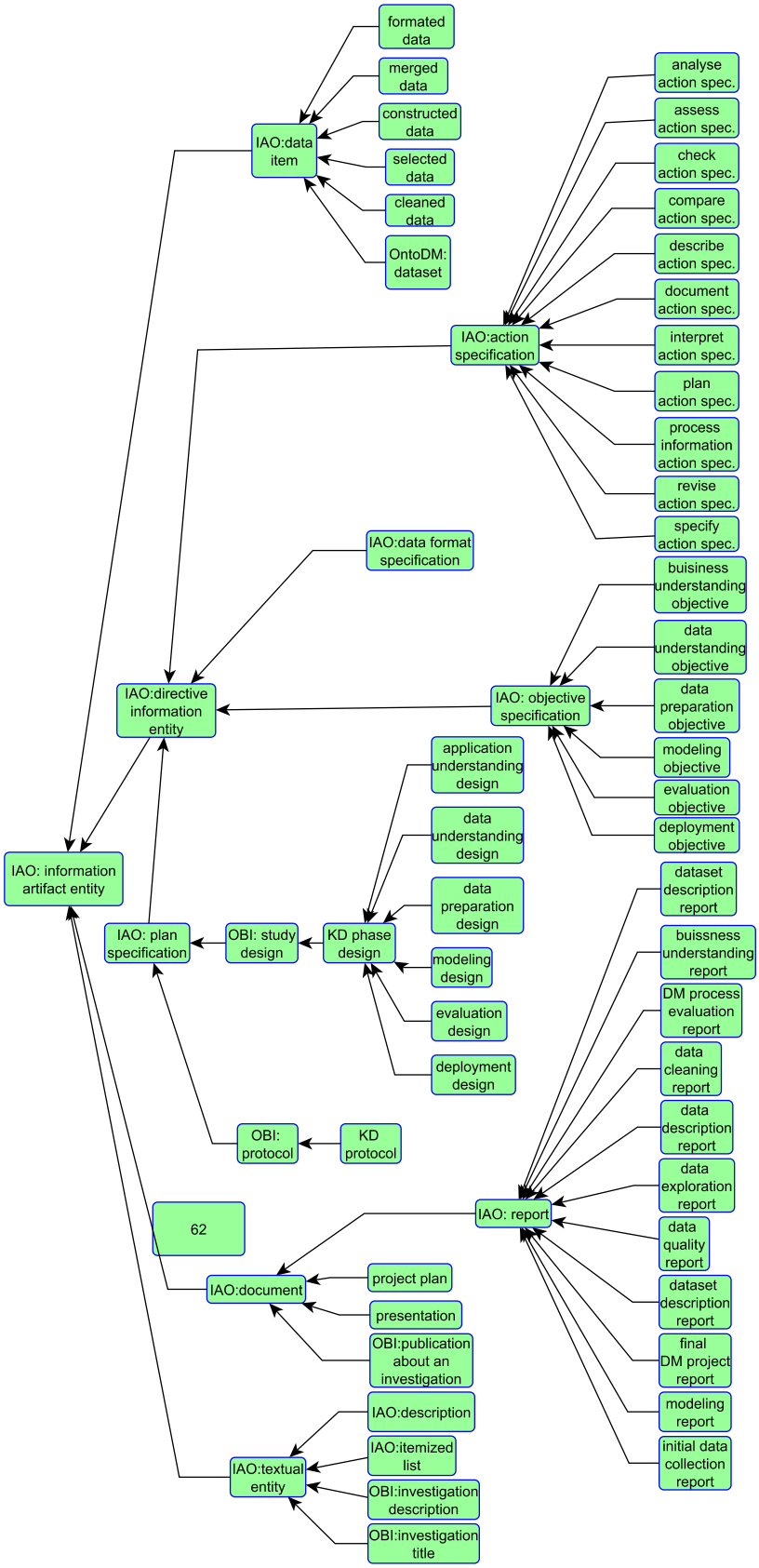


Figure 7.2: The specification level of OntoDM-KDD module.

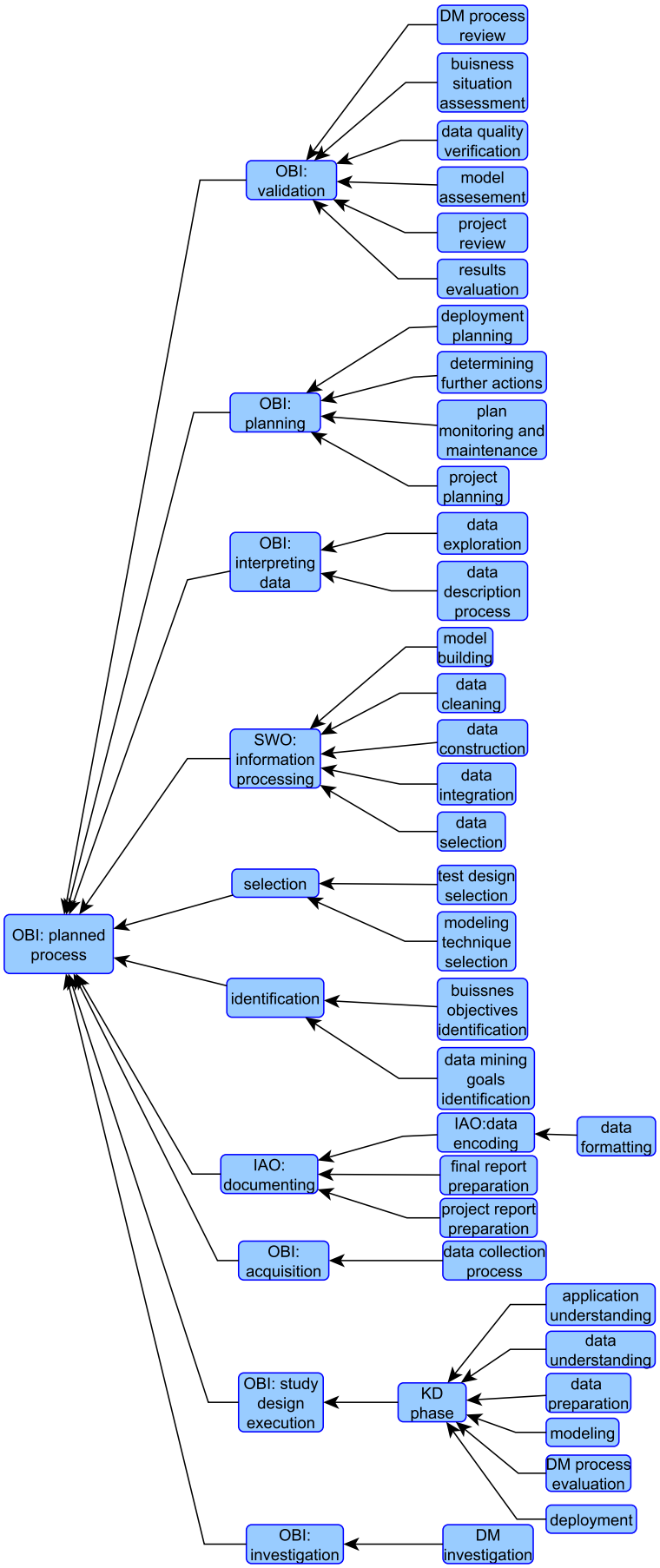


Figure 7.3: Upper level processual classes in OntoDM-KDD ontology.

7.3 Ontology module description

In this section, we first describe the representation a data mining investigation (Section 7.3.1). Next, we focus on each of the sub-processes in a DM investigation and discuss the most important representational issues that arise in the process of modeling them within the OntoDM-KDD ontology module. We first present the application understanding process (Section 7.3.2). Next, we discuss the representation of the data understanding process (Section 7.3.3) and data preparation process (Section 7.3.4). Furthermore, we present the modeling process (Section 7.3.5). Finally, we discuss the evaluation (Section 7.3.6) and deployment process (Section 7.3.7).

7.3.1 Data mining investigation

In the OntoDM-KDD ontology module, we define a *DM investigation* class as an extension of OBI *investigation* class (see Figure 7.4). An investigation has has as parts *planning* process, *documenting* process, and *study design execution* process. Furthermore, an investigation is characterized with an *investigation title*, *investigation description*, and *investigation identifier*. In addition, the investigation produces as output a *conclusion textual entity*. Finally, a *publication about an investigation* is a document about an investigation and it is an output of the documenting sub-process.

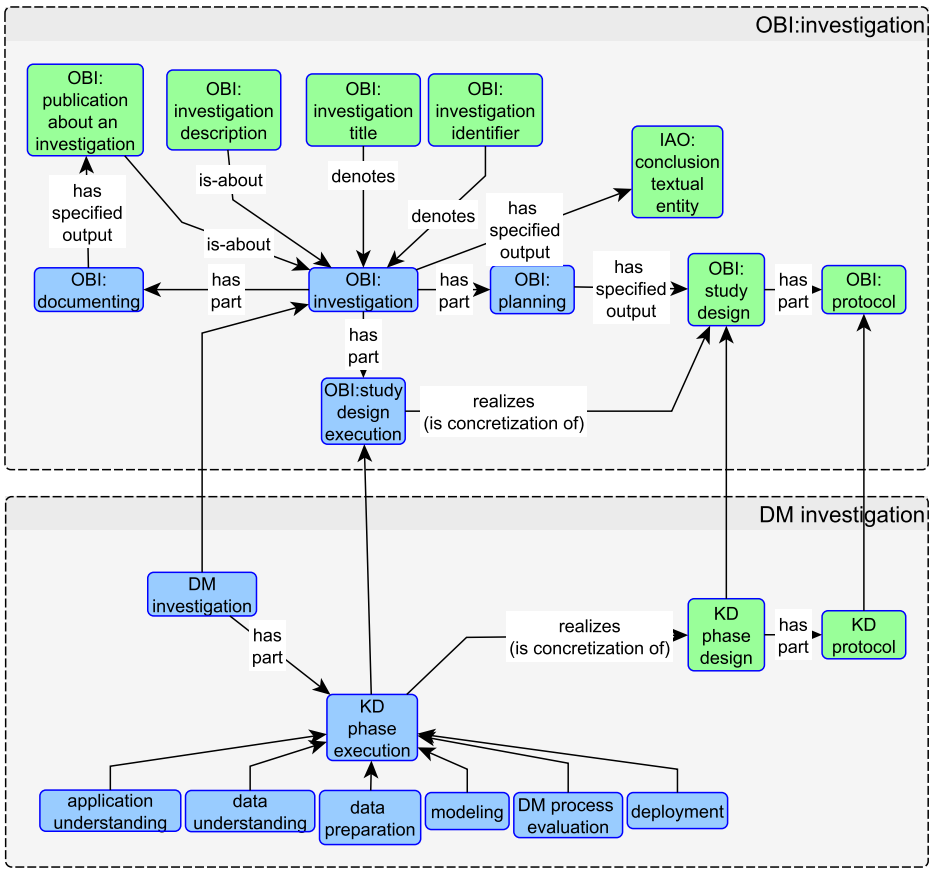


Figure 7.4: The data mining investigation class in OntoDM-KDD.

Furthermore, we extend the OBI *investigation* class with the *DM investigation* class (see Figure 7.4). We define a *DM investigation* as an *investigation* that has as its part *KD phase execution*, where this class is and extension of OBI *study design execution* and represents a realization of KD phase design that specifies each of the KD phases defined in CRISP-DM methodology. Each of the KD phases execution processes has as its part sub-processes as

defined in the CRISP-DM specification (see Figure 7.5), and they represent realizations of the KD protocols defined for each of the phases by a KD phase design specification. In this version of OntoDM-KDD, we can represent a sequential ordering of the KD phases by using the IS-PRECEDED-BY relation (see Figure 7.5).

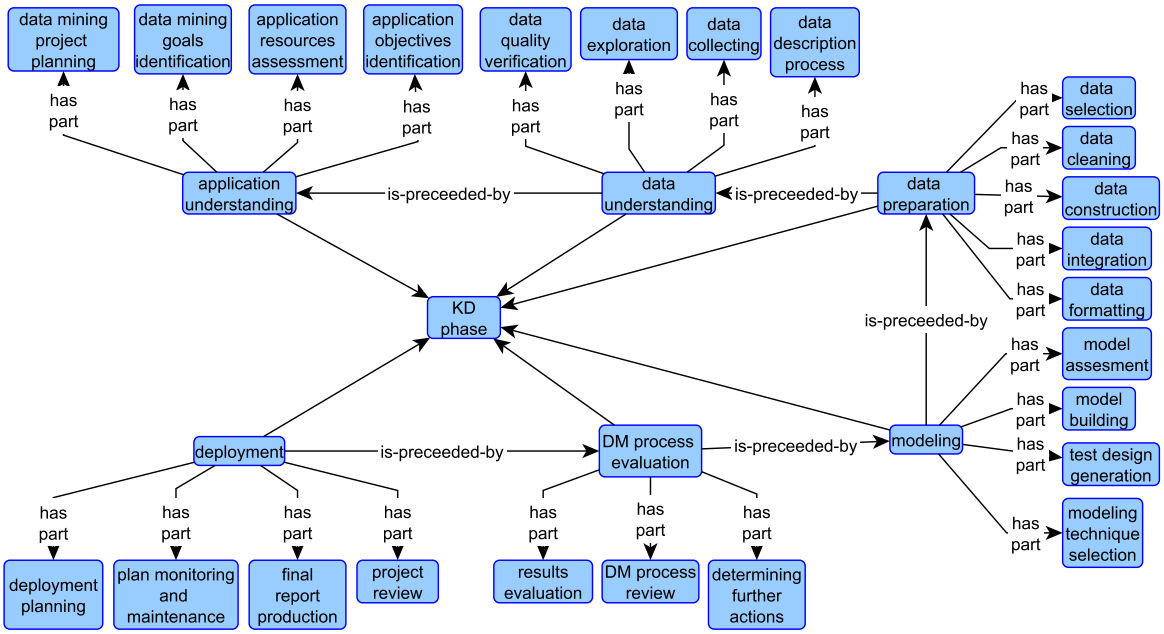


Figure 7.5: KD process and its sub-processes.

7.3.2 Application understanding process

The initial phase in a DM investigation focuses on identifying the of objectives and requirements of the investigation, from an application (or business) perspective. The goal of this process is to convert the knowledge about the application domain into a data mining problem definition and to generate a plan for achieving the application objectives. In this subsection, we describe the representation and the definition of the application understanding process in the context of the OntoDM-KDD ontology module (see Figure 7.6).

The *application understanding* class is a sub-class of *KD phase* and represents a planned process. In the ontological vocabulary, the *application understanding* process can be defined as a *KD phase* that realizes a concretization of an *application understanding design*, achieves the planned objective an *application understanding objective* and has specified output *application understanding report*. The *application understanding process* includes as parts four sub-processes. These include: *application objectives identification*, *application resources assessment*, *identification of data mining goals* and *generation of a project plan*.

The process of *application objectives identification* is a sub-class of the more general class of *identification* processes. In this process, a data analyst (active participant or agent) needs to identify in detail, from the application (or business) perspective, what are the objectives that need to be achieved by applying data mining to the application domain at hand. At the end of the process, the analyst needs to produce as output an *application background description*, an *application objectives description*, and an *application success criteria description*.

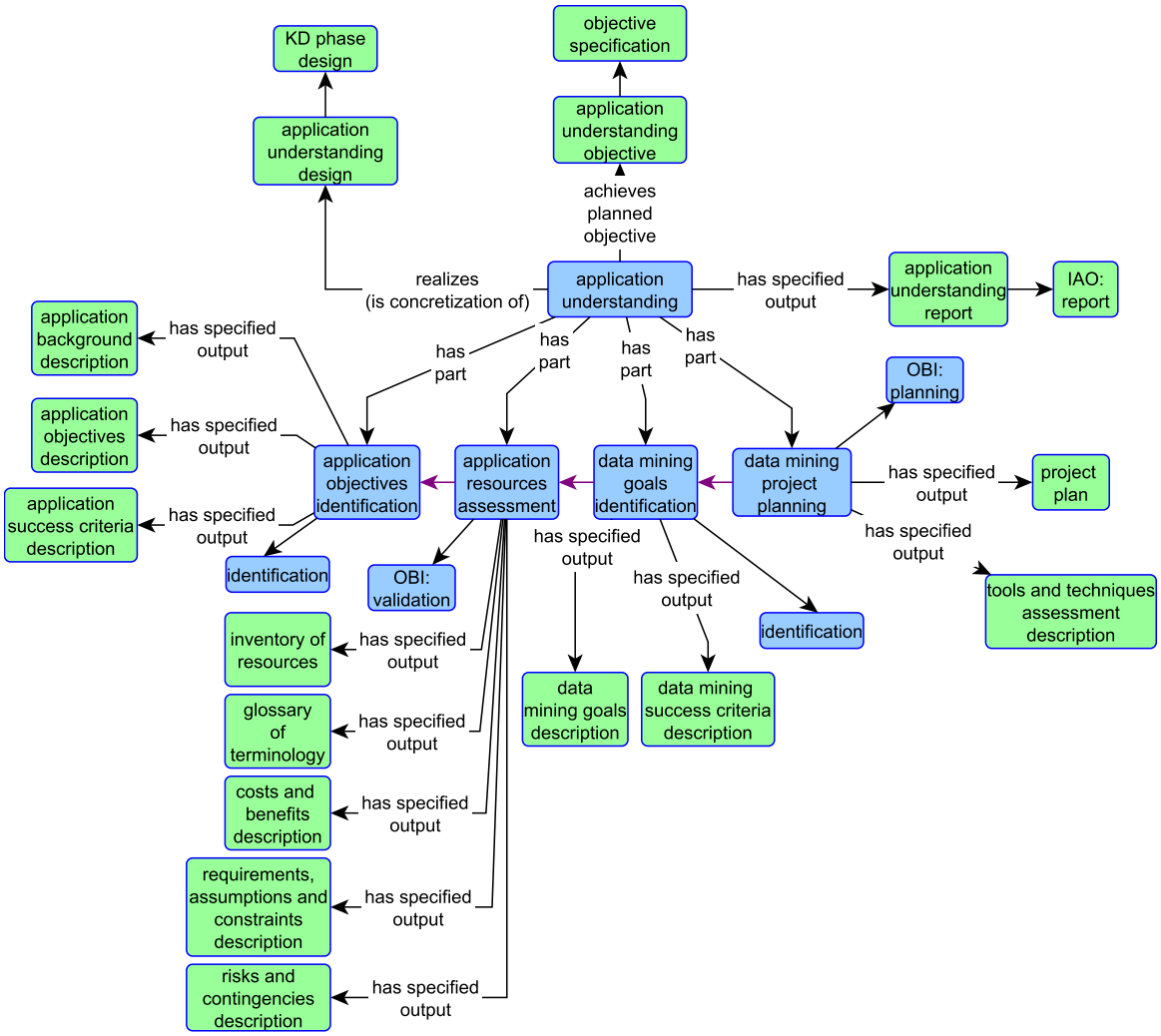


Figure 7.6: Application understanding process in OntoDM-KDD.

The process of *application objective identification* is a realization of a concretization of an *application objective identification protocol* (See Figure 7.7). The protocol, which is a part of the application understanding design specification, in its turn contains a specification of the actions that are realized in the process. For example, this process realizes actions such as *describe data mining problem*, *specify application success criteria*, *identify who assesses the success criteria* and others. These action specifications are subclasses of general classes of actions, such as *describe action*, *specify action* and *identify action*. In the OntoDM-KDD ontology module, we represent and provide action specifications for all processes in a similar way.

The process of *application resources assessment* is a sub-class of the OBI *validation* process. It involves the process of assessing the information about all resources, constraints, assumptions and other factors that need to be considered in order to determine the data mining goals and the project plan. The outputs of this process include: an *inventory of resources*, a *glossary of terminology*, *costs and benefits description*, a *requirements, assumptions and constraints description*, and a *risks and contingencies description*.

The process of *identification of data mining goals* is a sub-class of *identification* process. The objective of this process is to determine a specification of the data mining goals and establish a set of data mining success criteria in order to be able to evaluate the success of the data mining investigation at hand. The output of this process includes a *data mining goals description* and a *data mining success criteria description*.

The process of *data mining project planning* is a sub-class of the OBI *planning* process.

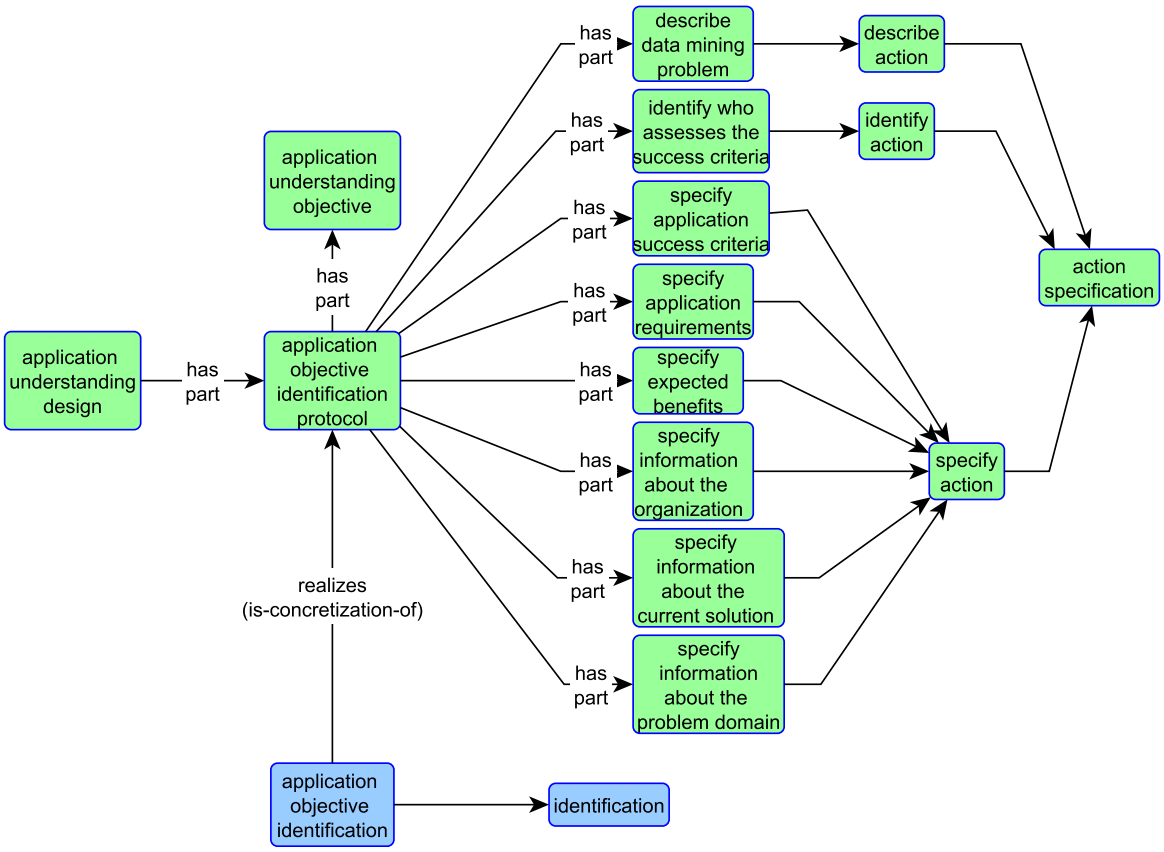


Figure 7.7: The process of objective identification in OntoDM-KDD.

The objective of this process is to produce a specification of a plan in order to achieve the data mining and application goals. The outputs of the process include a *project plan* and a *tools and techniques assessment description*.

7.3.3 Data understanding process

The data understanding process starts with an initial data collection, then proceeds with activities in order to get familiar with the data, to identify the data quality, discover first insights into the data, etc. In this subsection, we describe the representation and the definition of the data understanding process in the context of the OntoDM-KDD ontology module (see Figure 7.8).

The *data understanding* class is a sub-class of *KD phase* and represents a planned process. In the ontological vocabulary, the *data understanding* process can be defined as a *KD phase* that realizes a concretization of a *data understanding design*, achieves the planned objective *data understanding objective*, and has as parts four sub-processes: *data collection*, *data description process*, *data exploration*, and *data quality verification*.

The process of *data collection* is a subclass of the OBI *acquisition* process. The process includes the actions of acquisition and access to the project data, listed in the specification of the project resources. The output of the process includes an *initial data collection report*. The report lists the identified datasets, their location, methods used to obtain them and any encountered problems with the data acquisition.

The *data description* process is a subclass of the OBI *interpreting data* process. It includes the actions of examining the basic properties of the data acquired in the process of data collection. The output of this process includes a *data description report*. The report contains information about the format of the data, the quantity of the data, number of data records, etc.

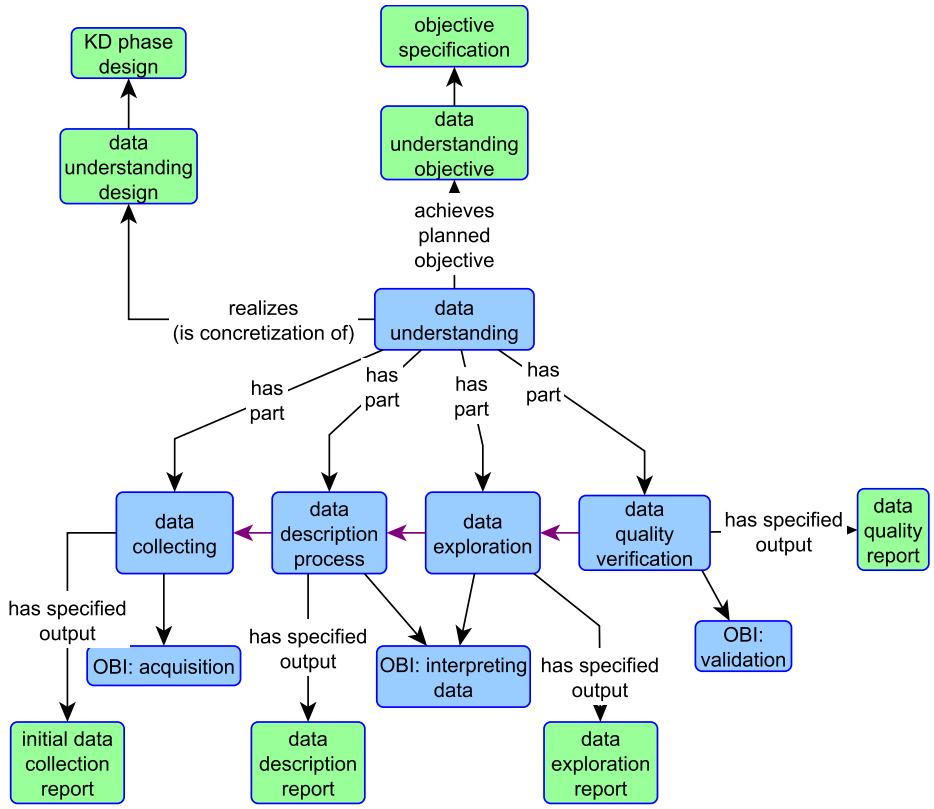


Figure 7.8: The data understanding process in OntoDM-KDD.

The *data exploration* process is also a sub-class of the OBI *interpreting data* process and includes actions of querying and visualization of the data in order to answer some of the data mining questions that can be addressed just by using those methods. Examples of such actions include: visualization of the data distribution, identifying relations between data attributes, performing simple statistical analysis, and others. The output of this process is a *data exploration report* that describes the initial findings resulting from the data.

The *data quality verification* is a sub-class of the OBI *validation* process and includes actions of examining the data quality. Examples of such actions include the assessment of data completeness, data correctness, whether the data contains errors and how common are the errors, whether there are missing values in the data, etc. The output of this process is a *data quality report*. The report contains the descriptions of the results of the data quality verification, a list of quality problems, and suggestions for possible solutions.

7.3.4 Data preparation process

The data preparation process covers the actions needed to construct a final dataset from the raw data. This dataset is to be used in the data mining (or modeling) phase. In this subsection, we describe the representation and the definition of the data preparation process in the context of the OntoDM-KDD ontology (see Figure 7.9).

The *data preparation* class is a sub-class of *KD phase* and represents a planned process. In the ontological vocabulary, the *data preparation* process can be defined as a *KD phase* that realizes a concretization of a *data preparation design*, achieves the planned the objective *data preparation objective*, and has as outputs a *dataset* and a *dataset description report*. This process has as parts five sub-processes: *data selection*, *data cleaning*, *data construction*, *data integration*, and *data formatting*.

The process of *data selection* is a sub-class of the SWO *information processing* process. In this process, the data analyst (the agent in the process) has to decide on the data to be

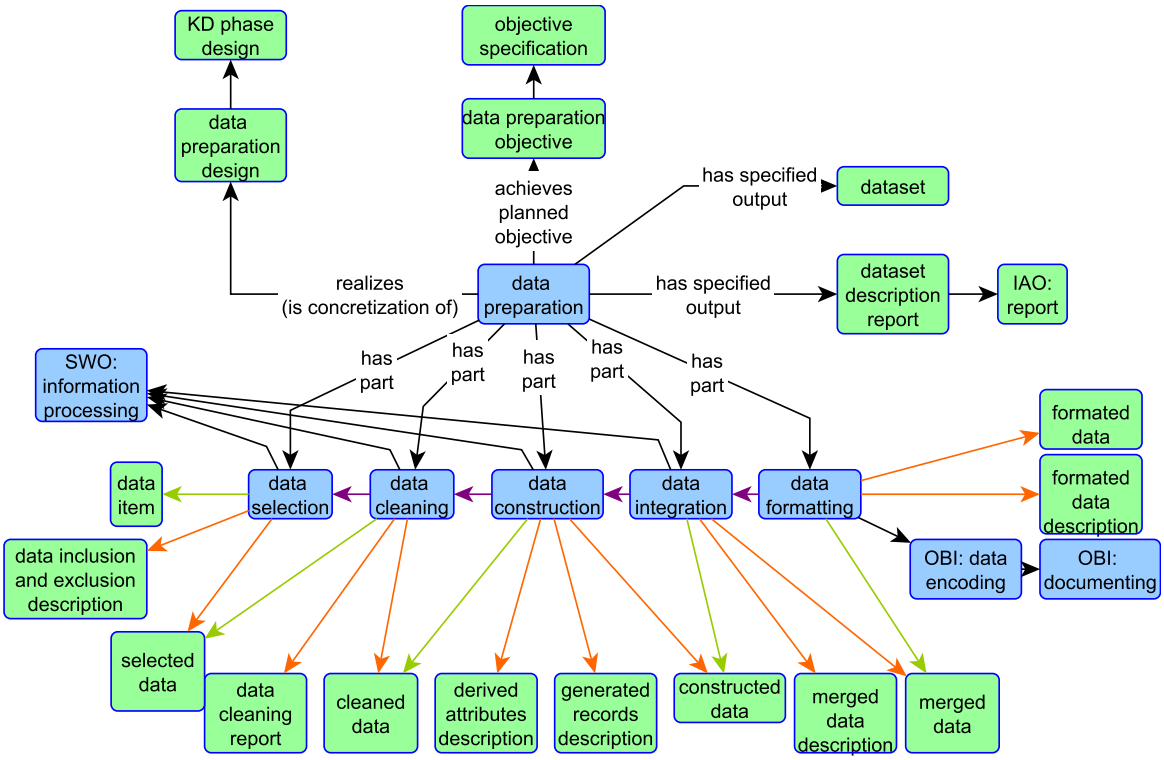


Figure 7.9: Data preparation process in OntoDM-KDD. The green arrows represent HAS-SPECIFIED-INPUT relation. The orange arrows represent HAS-SPECIFIED-OUTPUT relation. The violet arrows represent IS-PRECEEDED-BY relation.

used for analysis, having in mind data mining goals, data quality, and constraints on the quantity of available data. The data selection process involves actions such as selection of attributes and selection of data examples. The input to the process is some data item, i.e., raw data, while the output of this process is an *inclusion and exclusion description* and the *selected data*.

The process of *data cleaning* is a sub-class of the SWO *information processing* process and includes actions the data analyst has to perform in order to arise the data quality to the level required by the data mining algorithms. The actions include: the selection of clean subsets of data, the estimation of missing data by modeling, the insertion of values in the dataset, etc. The input to this process is the *selected data*. The output of this process is a *data cleaning report* and the *clean data*. In the report, the analyst lists the decisions and actions taken to address data quality problems and transformations of the data for cleaning purposes.

The process of *data construction* is a sub-class of the SWO *information processing* process and includes constructive data preparation actions. Examples of such actions include the generation of derived attributes, the generation of new data examples, and the transformation of values of attributes. The input to this process is the *clean data*, while the outputs include a *derived attributes description*, a *generated records description*, and the *constructed data*.

The process of *data integration* is a sub-class of the SWO *information processing* process. This process involves actions of applying the methods that combine and compose information from multiple sources (e.g., multiple databases) in order to create new data examples or values of attributes. The input for this task is the *constructed data*, while its outputs are the *merged data* and a *merged data description*.

The process of *data formatting* is a sub-class of the OBI *data encoding* class. This process involves actions for syntactical transformation of the data, while not changing its meaning,

for the purpose of using such data with a data mining algorithm that requires a specific format of the data. The input to this task is the *merged data*, while its outputs are the *formatted data* and a *formated data description*.

7.3.5 Modeling process

In the modeling process, various data mining algorithms are selected and applied to the dataset and their parameters are calibrated to optimal values. In this subsection, we give the representation and the definition of the modeling process in the context of the OntoDM-KDD ontology module (see Figure 7.10). The *modeling* class is a sub-class of *KD phase* and represents a planned process. In the ontological vocabulary, the *modeling* process can be defined as a *KD phase* that realizes a concretization of a *modeling design*, and achieves the planned objective *modeling objective*. This process has as parts four sub-processes: *modeling technique selection*, *test design generation*, *model building*, and *model assessment*.

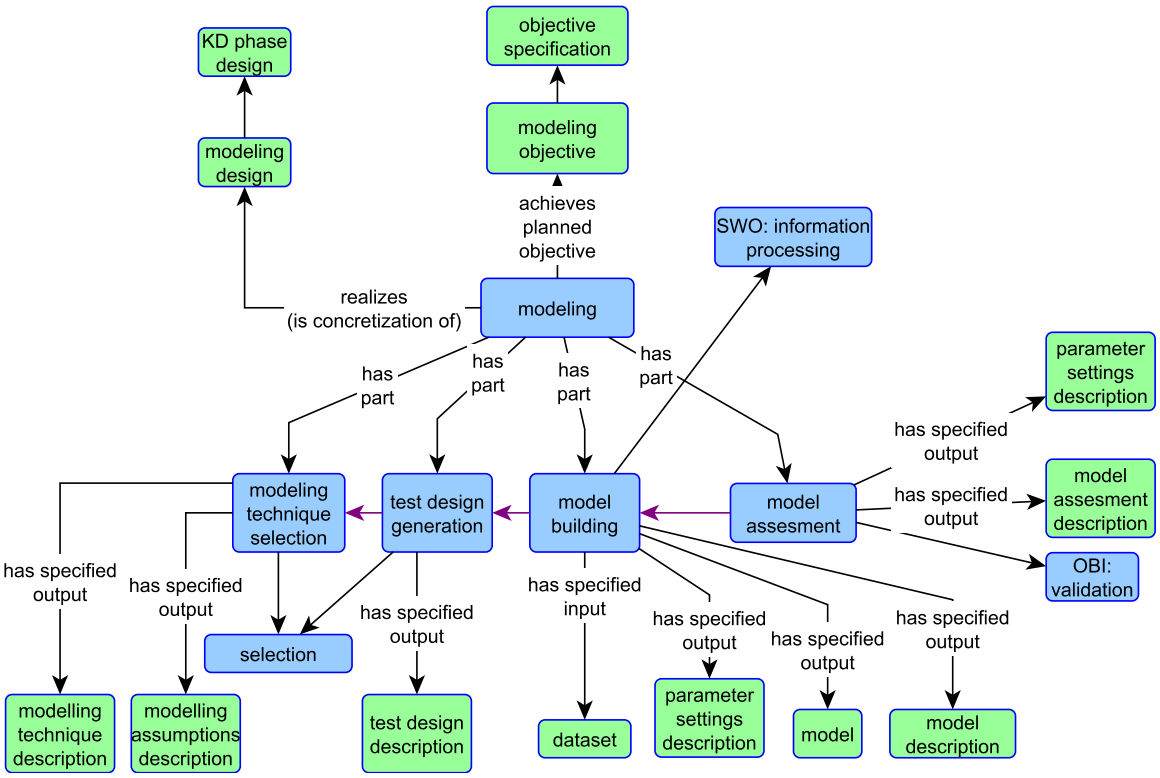


Figure 7.10: The modeling process in OntoDM-KDD.

The process of *modeling technique selection* is a sub-class of the *selection* process. This process involves the action of selecting a set of modeling techniques (or data mining algorithms) to be used for modeling (or data mining). The output of this task is a *modeling technique description*, which specifies the chosen modeling techniques to be used, and *modeling assumptions description* for each modeling technique separately.

The process of *generating a test design* is also a sub-class of the *selection* process. The process includes actions for specifying the procedure of testing the quality and validity of the models produced by applying the modeling techniques on a set of data. The output of this task is a specification of the test design with a plan for training, testing and evaluating the models including the procedure of how to divide a dataset into a training set, test set and validation set.

The process of *model building* is a sub-class of The SWO class *information processing*. This process involves the actions of application of a data mining algorithm on the prepared dataset to create a generalization. The input of this process is a *dataset*, obtained in the

process of data preparation. The outputs of this process include a *parameter settings description* for each algorithm separately, then the *generalization* produced by the algorithms, and finally a *model description*.

The process of *model assessment* is a sub-class of the OBI *validation* process. This process includes activities for interpretation of the generalizations produced in the task of model building by using domain knowledge, data mining success criteria, the desired test design, and the evaluation criteria. The outputs of this process include a *model assessment description* and a revised *parameter settings description* for the next iterative task of building generalizations, until generalization that satisfies the data mining success criteria is obtained.

7.3.6 DM process evaluation

At this stage in the data mining investigation, generalizations have been built that have high quality from the perspective of data analysis. The next important step is to evaluate and review the steps executed to construct the generalization in order to guarantee that it satisfies the application (or business) objectives. In this subsection, we give the representation and the definition of the data mining process evaluation in context of the OntoDM-KDD ontology (see Figure 7.11).

The *data mining process evaluation* class is a sub-class of *KD phase* and represents a planned process. A key objective of this process is to determine if all application objectives have been satisfied and to what extent, with the final goal to decide how to use the data mining results. In ontological vocabulary, the *data mining evaluation* process can be defined as a *KD phase* that realizes a concretization of a *DM process evaluation design*, achieves the planned objective *DM process evaluation objective*, and has as output a *DM process evaluation report*. This process has as parts three sub-processes: *results evaluation*, *DM process review*, and *determining further actions*.

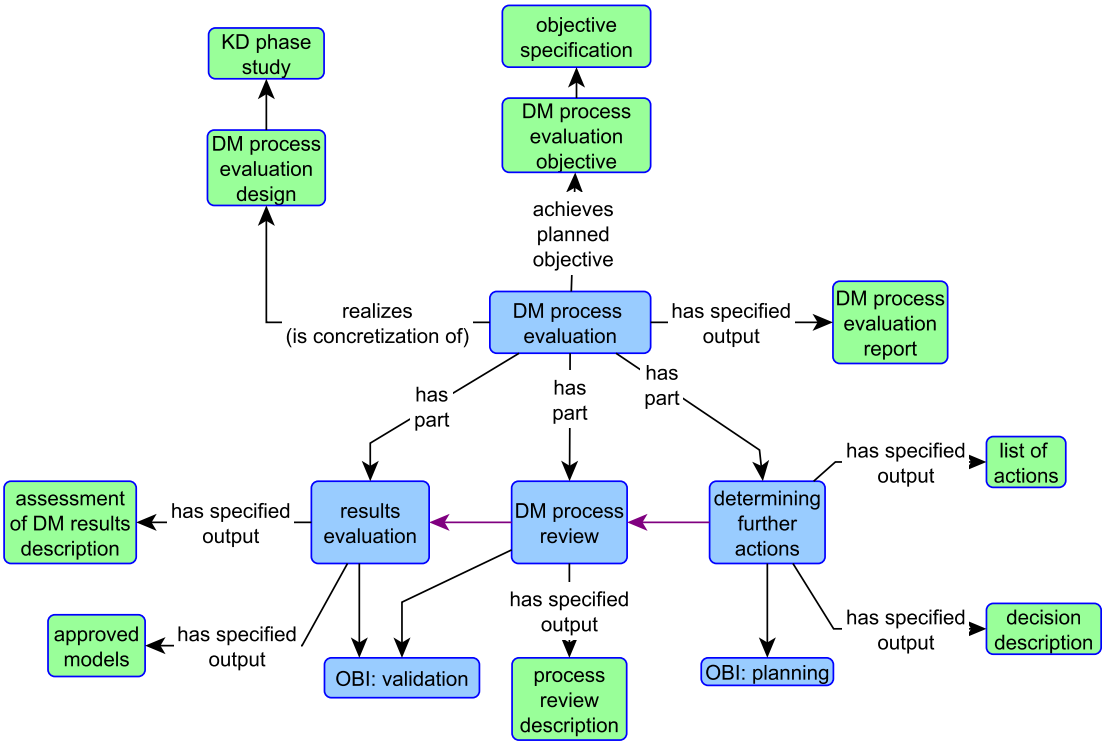


Figure 7.11: The class *data mining process evaluation* in OntoDM-KDD.

The process of *evaluating results* is a sub-class of the OBI *validation* process. It includes actions to assess the degree to which the produced generalization meets the application (or business) objectives: In contrast, unlike the last subprocess from the process of modeling

deals with the assessment of the produced generalizations from the perspectives of accuracy and generality of the generalization. Another part of the evaluation is to test the generalization on real test applications. The output of this process includes an *assessment of data mining results description* (with respect to application success criteria) and the *approved generalizations* (models).

The *DM process review* is a sub-class of the OBI *validation* process. It includes the actions to assess if the data mining engagement is performed adequately. This is done in order to determine if there is any important factor or task (in the data mining investigation) performed that has been overlooked. The output of this process includes a *process review description*. The description contains a summary of the review and highlighted activities that have been missed and the ones that need to be repeated.

The process of *determining further actions* is a sub-class of the OBI *planning* process. This process is performed in order to decide whether to finish the data mining investigation and move to the deployment or to perform additional iterations of other processes. It includes activities such as analysis of remaining resources and budget that influences decisions. The outputs of this process include a *list of actions* and *decision description*, the latter outlining how to proceed with the data mining investigation.

7.3.7 Deployment process

The creation of a generalization is usually not the end of a data mining investigation. Depending on the requirements and the application (or business) objectives, the deployment phase can variate from a simple report generation to a complex deployment, such as implementing the obtained generalization within some complex system. In this subsection, we describe the representation and the definition of the deployment process in context of the OntoDM-KDD ontology (see Figure 7.12).

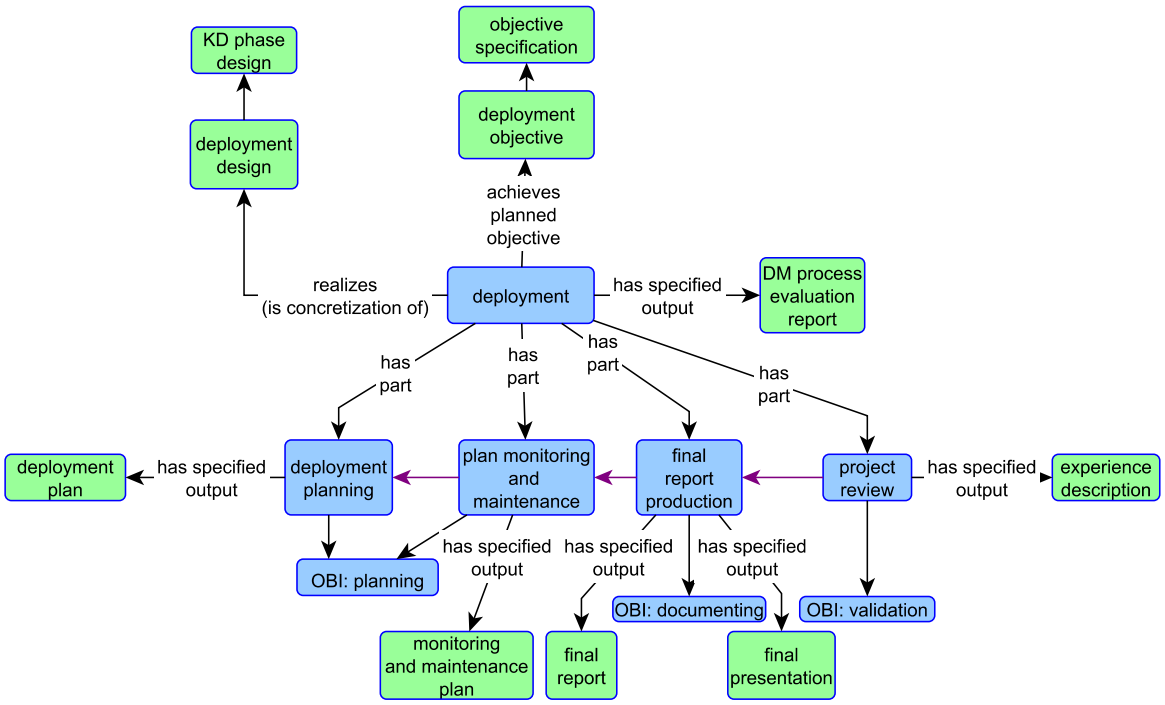


Figure 7.12: The class *deployment* in OntoDM-KDD.

The *deployment* class is a sub-class of *KD phase* and represents a planned process. In ontological vocabulary, the *deployment* process can be defined as a *KD phase* that realizes a concretization of a *deployment design* and achieves the planned objective *deployment objective*. This process has as parts four sub-processes: *deployment planning*, *plan monitoring*

and maintenance, *final report production*, and *project review*.

The process of *deployment planning* is a sub-class of the OBI *planning* process. The process involves actions for deployment of data mining results into the business application at hand. The output of this process includes a *deployment plan*, which summarizes the deployment strategy.

The process of *plan monitoring and maintenance* is a sub-class of the OBI *planning* process. It involves actions for a careful preparation of a maintenance strategy, to be taken into account to avoid long periods of incorrect usage of data mining results. The output of this process is a *monitoring and maintenance plan*, which summarizes the monitoring and maintenance strategy including the necessary steps and how to perform them.

The process of *final report production* is a sub-class of the OBI *documenting* process. It involves actions for producing a report with the summary of the data mining investigation, its results and experiences. The output of this process are the *final report* and the *final presentation*. The final report is a written report of the data mining investigation and it includes summary of all project deliverables and results. The final presentation is intended to present the project results at a meeting at the conclusion of the project.

The process of *project review* is a sub-class of the OBI *validation* process. It involves actions of assessing the completed data mining project by identifying the problems, summarizing the positive impacts, and giving suggestions on how to improve the problematic parts of the project. The output of this process is the *experience documentation* that summarizes important experiences made during the project.

8 Ontology evaluation

In the previous four chapters, we presented the design and implementation of the OntoDM ontology and its three modules OntoDT, OntoDM-core, and OntoDM-KDD. In this chapter, we address the issue of their evaluation. In general, there is still no commonly accepted practice for the evaluation of ontologies. Various approaches to the evaluation of ontologies are considered in the literature, depending on what kind of ontologies are being evaluated and for what purpose. First, ontologies can be evaluated by themselves or within some context. Next, ontologies can be evaluated within an application context (application based ontology evaluation)(Brank et al., 2005). Finally, ontologies can be evaluated in the context of an application and a given task (task based ontology evaluation) (Porzel and Malaka, 2004).

Gómez-Pérez (2004) separates the ontology evaluation procedure into two separate tasks: ontology verification and ontology validation. Ontology validation is the task of evaluating whether the ontology has been built correctly. Verification checks the encoding of the specification and detects errors such as circular class hierarchies, redundant axioms, inconsistent naming schemes, etc. Ontology verification confirms that the ontology has been built according to some specified ontology design criteria. Ontology validation, on the other hand, is the task of evaluating if the correct ontology has been built. Validation refers to whether the meaning of the definitions matches with the conceptualization the ontology is meant to specify.

In this thesis, we first present and discuss the set of ontology metrics used for the OntoDM ontology and its three modules (Section 8.1). Next, we assess how well the ontology meets a set of predefined design criteria and ontology best practices (discussed in detail in Obrst et al. (2007)) (Section 8.2). Furthermore, we assess the ontology towards set of competency questions (discussed in detail in Grüninger and Fox (1995)) (Section 8.3). Finally, we assess the OntoDM domain coverage by using a data mining topic ontology constructed in a semi-automatic fashion from a collection of articles from the domain (Section 8.4).

8.1 Statistical ontology metrics

The use of ontology metrics represents an important approach that can help to assess and qualify an ontology. There is a variety of ontology metrics available for assessing the ontologies (Garcia et al., 2010). In this thesis, we present the statistical ontology metrics used for the three ontology modules OntoDT, OntoDM-core, and OntoDM-KDD, which can be obtained from the Protégé software and the BioPortal web service. This includes metrics such as number of classes, number of individuals, maximum depth, average number of siblings, maximum number of siblings, sub-class axioms count, disjoint classes axioms count, and annotation assertion axioms count. The statistical ontology metrics are presented in Table 8.1.

From Table 8.1 we can see several interesting aspects about the proposed ontology. First, according to the number of classes and individuals we can see that the OntoDM-core is the best developed module from OntoDM, and also the most complex one if we take into consideration the maximum depth and the sub-class axiom count. Second, the other two

Table 8.1: Statistical metrics for the OntoDM ontology modules.

Metrics name	**OntoDT module**	**OntoDM core module**	**OntoDM KDD module**
Number of classes	144	760	264
Number of individuals	192	221	0
Maximum depth	1	20	1
Average number of siblings	7	9	11
Maximum number of siblings	21	28	25
Sub-class axioms count	375	1446	521
Disjoint classes axioms count	111	354	53
Class assertion axioms count	189	247	0
Annotation assertion axioms count	398	2871	553

modules (OntoDT and OntoDM-KDD) are less developed parts of the OntoDM which can be seen from the smaller number of classes and individuals. Next, from the annotation asseration axioms count we can see that all three ontology modules are heavily annotated with annotation properties such as comments, definitions, etc. Furthermore, we can also see from the disjoint classes axioms count that all three ontology modules contain large number of such axioms, in order to separate classes from one another as if not explicitly stated with an axiom, the child classes from the same parent are assumed to overlap.

8.2 Ontology assessment towards the design principles

In Chapter 4, we introduced and presented a set of predefined ontology best practices and design criteria that we intended to use for the development of the OntoDM ontology. After the ontology has been constructed, we can perform an assessment of the ontology towards these design principles and ontology best practices in order to see how the finalized ontology fits them. First, we perform scope and structural assessment (Section 8.2.1). Second, we perform naming and vocabulary assessment (Section 8.2.2). Next, we perform documentation ans collaboration assessment (Section 8.2.3). Finally, we conclude with availability, maintenance and use assessment (Section 8.2.4).

8.2.1 Scope and structure assessment

The scope and structure assessment involves assessment of the OntoDM ontology towards a set of design criteria and ontology best practices that constrain the scope of the ontology and its structure. This includes: coverage, upper-level ontology, relations, reuse of other ontologies, modularity, use of disjoint classes, use of single inheritance, IS-A completeness, established domains and ranges of the relations, use of inverse relations, orthogonality with other ontologies, and instantability. Table 8.2 summarizes the OntoDM assessment towards this set of principles.

From Table 8.2 we can see that first the OntoDM is general enough to represent mining of structured data and provides coverage of the complete data mining process for the purpose of representing data mining investigations. The ontology uses a upper-level ontology BFO, a set of formal relations, reuses other ontological resources, and it is modular. Regarding the structure of the ontology, OntoDM extensively uses disjoint classes, which is evident from statistical metrics presented in Table 8.1. Furthermore, by inspecting the ontology we can see that it is IS-A complete and it uses only single inheritance of classes to reduce possible inconsistencies and errors in the reasoning process. Since all the relations are formally

Table 8.2: Scope and structure assessment.

#	**Principle**	**Assessment**
1	coverage	OntoDM provides a representation of data mining investigations, and it is general enough to represent the mining of structured data.
2	upper-ontology	OntoDM uses the BFO ontology as an upper-level ontology.
3	relations	OntoDM uses relations defined in RO, IAO, EXACT and OBI. The relations defined in IAO and OBI are candidates for inclusion into RO.
4	ontology reuse	OntoDM reuses classes and relations from OBI, IAO, SWO and EXACT.
5	modularity	OntoDM is composed of three ontology modules: OntoDT, OntoDM-core, and OntoDM-KDD.
6	use of disjoint classes	In OntoDM, we extensively use disjoint class axioms.
7	use of single inheritance	In OntoDM, each class has only one superclass. This reduces the potential inconsistency and errors in reasoning processes.
8	IS-A completeness	All the OntoDM classes are connected via the IS-A relation. There are no orphan classes.
9	domains and ranges for relations	Imported relations from RO, IAO and OBI have defined ranges and domains.
10	inverse relations	Most of the imported relations from RO, OBI, and IAO have defined inverse relations.
11	orthogonality with other ontologies	OntoDM is orthogonal to other ontologies already lodged within OBO.
12	instantability	Some classes from OntoDM have defined instances. In future work, more extensive population of the ontology with instances is planned.

defined, all of them have defined domains and ranges and most of them have defined inverse relations. Finally, OntoDM is orthogonal to other ontologies already in OBO.

8.2.2 Naming and vocabulary assessment

The naming and vocabulary assessment involves assessment of the OntoDM ontology towards a set of design criteria and ontology best practices dealing with naming entities in the ontology. This includes: the ontology language, the use of annotation properties, annotation of labels of classes and relations, defining the ontology namespace, naming ontology terms, multi-lingual capabilities, the use of naming conventions, and referencing of external classes. Tab. 8.3 summarizes the OntoDM assessment towards this set of principles.

From Table 8.3 we can first see that OntoDM is expresses in OWL-DL ontology language, which represents a standard in ontology representation. Second, the ontology reuses OBI consortium meta-data in order to provide heavy annotation of classes and relations, which is evident from the statistical metrics from Table 8.1. Furthermore, for all classes and relations we provide label annotations to provide human readable names. The ontology has its defined namespace and it uses a recommended format for ontology terms, which is composed of the module name and five digit code. Finally, the ontology follows OBO Foundry naming conventions, uses the MIREOT principle for referring the external classes, and at this phase of development still does not provide multi-lingual capabilities.

Table 8.3: Naming and vocabulary assessment.

#	Principle	Assessment
1	ontology language	OntoDM is expressed in the W3C standard Web Ontology Language OWL-DL.
2	use of annotation properties	We reuse the OBI consortium defined meta-data[1] to provide additional semantic annotation of the classes and relations
3	label annotations	We use label annotations to provide human readable names of classes and relations in the ontology.
4	ontology namespace	The ontology has its own namespace `http://kt.ijs.si/panovp/OntoDM#`. The classes and relations that are imported from other ontologies have kept their source ontology namespace and ID.
5	ontology term IDs	The IDs of the ontology terms are diferent for each ontology module and include a combination of an ontology module ID and a five digit code: `OntoDT_xxxxx` for the OntoDT module, `OntoDM_xxxxx` for the OntoDM-core module, and `OntoDM-KDD_xxxxx` for the OntoDM-KDD module.
6	multi-lingual capabilities	At this moment the OntoDM ontology does not provide multi-lingual capabilities.
7	naming conventions	The ontology uses set of naming conventions provided by the OBO Foundry.
8	referencing external classes	The external classes are referenced by using the MIREOT principle.

8.2.3 Documentation and collaboration assessment

The documentation and collaboration assessment involves assessment of the OntoDM ontology towards a set of design criteria and ontology best practices dealing with documenting the ontology and its inclusion into collaborative efforts. This includes: providing definitions of classes and relations, documenting the ontology, and collaboration efforts. Tab. 8.4 summarizes the OntoDM assessment towards this set of principles.

Table 8.4: Documentation and collaboration assessment.

#	Principle	Assessment
1	definitions	Most of the OntoDM classes have textual definitions. They are regularly updated and revised.
2	documentation	The ontology is documented on its dedicated web page and in journal, conference and workshop publications.
3	collaboration efforts	OntoDM participates in the Data Mining Ontology (DMO) Foundry with the aim of developing a core data mining mid-level ontology.

From Table 8.4 we can see that most of the OntoDM classes have definitions. Second, the ontology is well documented on a dedicated web page. Regarding collaboration efforts, it is a participant in the DMO Foundry with the aim of developing a core DM ontology at the mid-level.

8.2.4 Availability, maintenance and use assessment

The availability, maintenance and use assessment involves assessment of the OntoDM ontology towards a set of design criteria and ontology best practices dealing with testing the ontology, its availability and use. This includes: use of reasoners, openness and availability, versioning, established users of the ontology, maintenance, and handling of obsolete classes. Tab. 8.5 summarizes the OntoDM assessment towards this set of principles.

Table 8.5: Availability, maintenance and use assessment.

#	**Principle**	**Assessment**
1	use of reasoners	To test the class and relations consistency we use the Pellet reasoner.
2	openness and availability	OntoDM is open and is available in its web page `http://www.ontodm.com` and additionally at BioPortal (`http://bioportal.bioontology.org/`).
3	versioning	For tracking the changes in the ontology we use the industry standard Subversion tool
4	users of the ontology	The ontology has established a set of users. The ontology is reused by Expose ontology, and some classes are imported by the SWO ontology.
5	maintenance	The ontology has a dedicated person that cares about its maintenance.
6	handling of obsolete classes	Deleted classes in the OntoDM class hierarchy are listed under the *obsolete* class, so that applications based on them can still use the terms. The domain terms that have been collected so far but are still not represented in the ontology are listed under the *non-curated* class in the OntoDM class hierarchy.

From Table 8.5 we can first see that the OntoDM ontology uses the Pellet reasoner for testing class and relation consistency. Second, the OntoDM ontology is open-source and available on a dedicated web page and on community portals. Next, the ontology uses a versioning tool to track changes and the ontology has an established set of users, which is evident from the use case presented in Section 9.5. Finally, the OntoDM ontology has a dedicated person for maintenance and has a mechanism for handling obsolete classes

8.3 Ontology assessment towards competency questions

Grüninger and Fox (1995) proposed a methodology for the design and evaluation of ontologies. The methodology proposes to first define the ontology's requirements in the form of questions that the ontology must be able to answer, named ontology competency questions. Next, the terminology of the ontology and its classes and relations should be defined. The competency questions in this phase represent informal competency questions, since they are still not expressed in the language of the ontology. Once the informal competency questions have been posed for the new ontology, then the terminology of the ontology (its classes and relations) are specified using some first order logical language (e.g., description logics (Baader et al., 2003)). This language must provide the necessary terminology to formally restate the informal competency questions. For every informal competency question, there must be classes, instances and relations in the ontology, which are intuitively required to answer the question. Furthermore, having the language of the ontology, the competency questions are defined formally as an entailment with respect to the axioms in the ontology. Every newly developed ontology must be accompanied by a set of formal competency ques-

tions. This represents a way how to evaluate the ontology and claim that it is adequate. Finally, ontologies can be distinguished by the competency questions which they are capable of answering.

For the case of the OntoDM ontology, presented in this thesis, we established a list of informal competency questions, in the design phase, for all three ontology modules OntoDT, OntoDM-core, and OntoDM-KDD (see Tables 5.1, 6.1, and 7.1 respectively). Having build all three ontology modules and expressed them in a formal ontology language based on description logics (OWL-DL), we can formulate the formal competency questions for each of the modules and show that the ontology module is capable of solving and answering the questions it is designed to answer. For that purpose, we formulate the questions using the SPARQL-DL query language[2] (Sirin and Parsia, 2007), where the SPARQL-DL is a query language for querying OWL-DL ontologies. SPARQL-DL is a subset of the SPARQL language[3]. In Table 8.6, we show how the competency questions for the OntoDM-core module can be formulated in the language of the ontology using the SPARQL-DL language. The same can be shown for the OntoDT and the OntoDM-KDD modules.

Table 8.6: Formalization of the OntoDM-core competency questions using the SPARQL-DL language.

Q_n	**Query**	**SPARQL-DL representation**

1	Which datasets have data belonging to a datatype X?	`Q1(d):-Type(?d,dataset),`
		`PropertyValue(?ds,is-about,?d),`
		`PropertyValue(?ds,has-part,?datas),`
		`PropertyValue(?datas,is-about,x),`
		`Type(x,datatype).`

2	Which data mining tasks can be formulated on data of datatype X?	`Q2(dmt):-Type(?dmt,data_mining_task),`
		`PropertyValue(?dmt,has-part,?datas),`
		`PropertyValue(?datas,is-about,x),`
		`Type(x,datatype).`

3	Which data mining tasks can be formulated on a dataset X?	`Q3(dmt):-Type(?dmt,data_mining_task),`
		`PropertyValue(?dmt,has-part,?datas),`
		`PropertyValue(?ds,has-part,?datas),`
		`PropertyValue(?datas,is-about,x),`
		`Type(x,dataset).`

4	Which data mining algorithms solve a data mining task X	`Q4(dma):-Type(?dma,data_mining_algorithm),`
		`PropertyValue(?dma,has-part,x),`
		`Type(x,data_mining_task).`

5	Which data mining algorithms are applicable to data of datatype X?	`Q5(dma):-Type(?dma,data_mining_algorithm),`
		`PropertyValue(?dma,has-part,?dmt),`
		`PropertyValue(?dmt,has-part,?datas),`
		`PropertyValue(?datas,is-about,x),`
		`Type(x,datatype).`

continued on next page

[2]SPARQL-DL: `www.w3.org/2001/sw/wiki/SPARQL-DL`
[3]SPARQL-DL: http://www.w3.org/TR/rdf-sparql-query/

6	What is the set of the possible generalization types that are given as output by solving a data mining task X on data of type Y?	`Q6(gs):-Type(?gs, generalization_specification), PropertyValue(?gs,has-part,?datas), PropertyValue(x,has-part,?datas), Type(x,data_mining_task), PropertyValue(?datas,is-about,y), Type(y,datatype).`
7	What is the set of implementations for a DM algorithm X that have a parameter Y?	`Q7(dmi):-Type(?dmi,data_mining_algorithm_implementation), PropertyValue(?dmi,is-concretization-of,x), Type(x,data_mining_algorithm), PropertyValue(?dmi,has-quality,y), Type(y,parameter).`
8	What is the set of all generated generalizations on a dataset X?	`Q8(g):-Type(?g,generalization), PropertyValue(?g,is-specified-output-of,?dmae), PropertyValue(?dmae,has-specified-input,x), Type(x,dataset).`
9	Which DM software toolkit contains an implementation of a DM algorithm X?	`Q9(dmst):-Type(?dmst,data_mining_software_toolkit), PropertyValue(?dmst,has-part,?dms), PropertyValue(?dms,is-about,?dmi), PropertyValue(?dmi,is-concretization-of,x), Type(x,data_mining_algorithm).`
10	On which computer has a specific data mining algorithm X been executed?	`Q10(c):-Type(?c,computer), PropertyValue(?c,is-agent-of,?dmae), PropertyValue(?dmae,is-realized-by,?dmo), PropertyValue(?dmo,is-concretization-of,x), Type(x,data_mining_algorithm).`
11	What is the parameter setting of a given execution of a data mining algorithm X on a computer Y, having as input a dataset Z?	`Q11(ps):-Type(?ps,parameter_setting), PropertyValue(?dmo,has-information,?ps), PropertyValue(x,realizes,?dmo), Type(x,data_mining_algorithm_execution), PropertyValue(?dmae,has-agent,Y), Type(y,data_mining_algorithm), PropertyValue(?dmae,has-specified-input,z), Type(z,dataset).`

continued on next page

12	What is the set of datasets on which a given generalization X has been executed?	`Q12(dmst):-Type(?d,dataset),`
		`PropertyValue(?ge,has_specified_input,?d),`
		`PropertyValue(?ge,realizes,?g),`
		`Type(g,generalization).`

8.4 Evaluation of domain coverage using a data mining topic ontology

In this section, we present the evaluation of the domain coverage of the OntoDM ontology by comparing it with the semi-automatically constructed data mining topic ontology (briefly described in Section 3.3.2) constructed from a large corpus of data mining papers, using a tool for semi-automatic ontology construction OntoGen (Fortuna et al., 2007) (See Section 3.3.2). The data mining topic ontology was initially constructed in the context of the e-LICO project[4]for assessment of the Data Mining Optimization (DMOP) (see Section 3.2.4) ontology developed in the project, and was published in a report by (Vavpetič et al., 2011).

In the remainder of this section, we first summarize the process of construction of the topic ontology (Section 8.4.1). Next, we present the constructed data mining topic ontology (Section 8.4.2). Furthermore, we present the assessment of the OntoDM ontology with respect to the topic ontology (Section 8.4.3). The main part of the assessment is to identify the topics which are not covered by the OntoDM-core ontology, and are covered by the topic ontology. As a result, we list the main topics that are not represented in the OntoDM and compare the two ontologies.

8.4.1 Construction of the topic ontology

A topic ontology is a set of topics connected with different types of relations. In this report, the terms topic, concept and cluster will be used interchangeably, denoting a set of documents on a given topic. For large enough corpora, the construction of a topic ontology can be very time consuming. Semi-automated construction can be supported by using specialized tools, such as OntoGen (Fortuna et al., 2007), used in this work. First, the process of learning a topic ontology involves generation and preprocessing of a corpus of papers from the domain of interest. The output corpus should be free of duplicates, noise and in a correct format required by the software. Next, a topic ontology can be constructed using the semi-automatic ontology editor OntoGen.

The process of corpus generation is initiated by manually collecting and choosing a set of web sites and other sources that contain a reasonable amount of data mining and machine learning publications (documents) to be crawled. Since the corpus had to be unbiased, websites that focus on covering of only specific sub-fields of data mining should be avoided. In this case the sources include papers from ECML/PKDD conferences, ICML conferences, JMLR papers, ACM SIGKDD papers, PASCAL NoE collection of papers, and SIAM data mining conference. The list of used sources and the physical locations is provided by Vavpetič et al. (2011).

The next step in the corpus generation included web crawling of the documents available at the source sites by using the 'HTTrack' crawling program[5]. Additional care was taken to classify the gathered papers by the year of publication. In some cases, the whole conference paper set was obtained as a single PDF document containing the complete proceedings. These files had to be manually split into separate article files. Through the process of crawling 6350 documents were gathered. During the crawling phase it became evident that

[4]e-LICO project: `http://www.e-lico.eu/`
[5]HTTrack: `http://www.httrack.com/`

we may have obtained multiple copies of some papers via various sources. Approximately 220 duplicate papers were removed and we ended with 6,134 distinct PDF documents. Vavpetič et al. (2011) give the distribution of the documents per year and per source. Next, the PDF documents were converted into text files. For some documents it was not possible to obtain the textual converted version and they were filtered out. In addition, documents that were not written in the English language were excluded as well. Finally, manual inspection was performed and a final 5885 document corpus was produced at the end.

In order to use the tool for topic ontology construction Ontogen, the data needed to be additionally prepared in the named-line document format. The named-line document format consists of a list of documents separated by the new line character. Each line represents one document and the first word of the document is considered as its title. Due to the size of the corpus, only the abstract of each paper was used, as the amount of data was simply too large for sensible use of the OntoGen tool.

The two main design goals of the semi-automatic ontology construction tool OntoGen (Fortuna et al., 2007) are the visualization and exploration of existing entities from the ontology, and the addition of new entities or modification of existing concepts using simple and straightforward machine learning and text mining algorithms. The Ontogen system is an interactive tool that aids the user's input during the topic ontology construction process. It suggests concepts, relations between the concepts, and concept names, automatically assigns instances to the concepts, visualizes instances within a concept and provides a good overview of the ontology to the user through concept browsing and various kinds of visualizations. Finally, the assistance that the system provide to the user is based on the input data corpus. The data corpus affects the structure of the domain for which the user is building the topic ontology. The data is usually a document corpus, where ontological instances are either documents or named entities occurring in the documents. The system supports automated extraction of instances (used for learning concepts) and co-occurrences of instances (used for learning relations between the concepts) from the data.

All the documents from the input file were imported into OntoGen as a root concept which had to be divided into several sub-concepts. OntoGen treats each line of the input file as a document (in this case it is an abstract of the original document). The concepts can be extracted from the documents by using k-means clustering, visualization or active learning with Support Vector Machines (SVM):

- The k-means clustering algorithm (MacQueen, 1967) is used on the documents to induce several candidate concepts for the expansion of the ontology at each level of the topic hierarchy. Different values of k (the number of desired clusters) had to be experimented with to achieve satisfiable results, i.e. compact clusters of documents. By examining the keywords which represent the given concept, one can derive an appropriate compact cluster. Each of the concepts was divided further from the root concept separately by selecting such a value of the parameter k, that the concept included the documents that belong only to that concept and no other concepts included the keywords that describe this subconcept. When such a value of k is found for the first selected subconcept, the process is repeated on the remaining documents.

- Visualization is an alternative method to the k-means clustering. For this purpose, we can use the OntoGen's visualization tool. This visualization tool plots all the documents belonging to a concept to help the user identify further sub-concepts.

- Active learning using SVM can be used for devising sub-concepts. This is a supervised learning method based on the Support Vector Machines (SVM) algorithm (Cortes and Vapnik, 1995) and using the active learning approach. For this method the user enters a query and at each step the learning system asks the user to indirectly provide labels for the documents by answering questions whether a documents belongs to a given concept

or not. After some initial labeled sample is collected, the system displays additional information about the concept. The user can at this point continue answering the questions or end the process of active learning. The more questions the user answers, the more correct is the assignment of instances to the final concept, as the user provides more learning instances to the process of active learning.

8.4.2 Data mining topic ontology

Using the prepared document corpus and the Ontogen tool, we address the task of constructing a data mining topic ontology in a semi-automatic fashion, by iteratively building the IS-SUBTOPIC-OF taxonomy, splitting each concept into sub-concepts. In this way the number of documents per concept gradually decreases to a point where it is hard to extract further concepts in a sensible fashion. At this point, we have extracted a topic ontology of concepts which naturally stems from the given document corpus. The resulting topic ontology is shown in Figure 8.1.

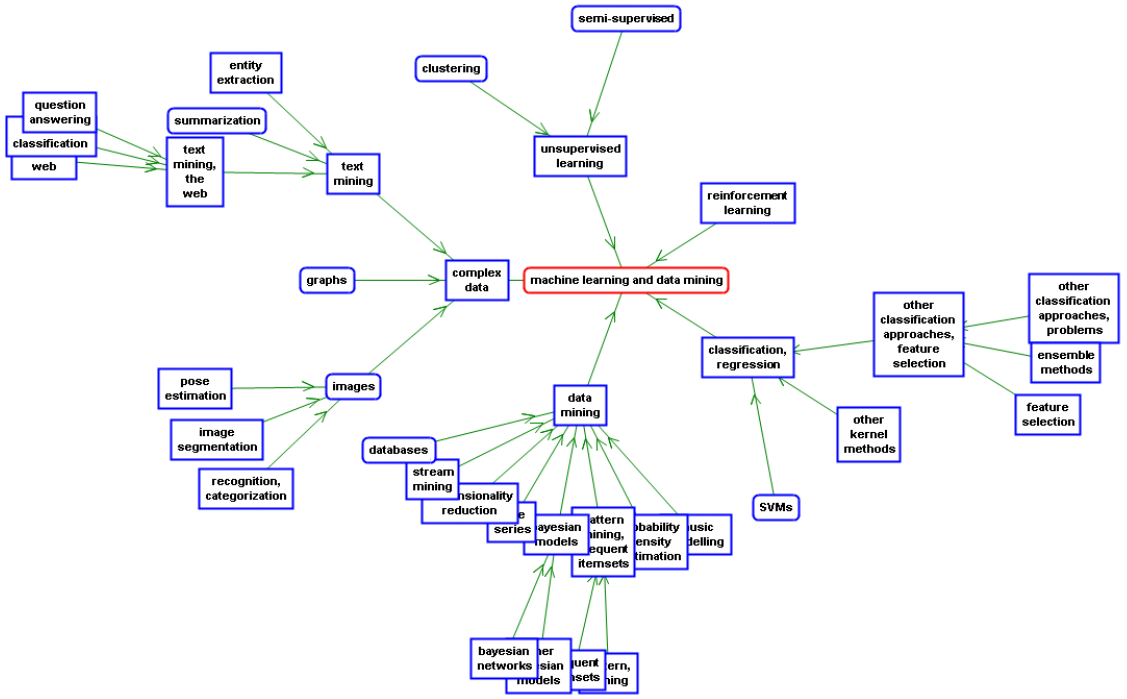


Figure 8.1: Data mining topic ontology obtained directly from the document corpus using only k-means clustering. The figure shows the output of the Ontogen software.

After analyzing the initial topic ontology, we concluded that the construction process needs domain expert input in order to obtain a more consistent and compact topic structure. By inspecting the concepts, different types of topics were identified. Some topics represented certain data types (e.g., images, graphs), some represented applications (e.g. bioinformatics) and some represented data mining tasks (e.g., classification) at the same level in the taxonomy. These were then manually moved and grouped under common super-concepts in order to achieve a better structure of the ontology. The final constructed semi-automatically is shown in Figure 8.2.

At a root concept we have machine learning and data mining, which is our topic domain of interest: all documents belong to this domain. At the first level, we distinguish between: supervised learning, unsupervised learning, reinforcement learning, complex data, inductive logic programming, data preprocessing, pattern mining and frequent itemsets and other data mining approaches. This structure is very much in line with the structure of most of the textbooks on machine learning and data mining. Each of the concepts at the first level

is additionally refined at the lower levels. For example, complex data is decomposed into the following sub concepts: graph, image, music modeling, stream mining, time series and text. This ontology of topics is in line with the defined tasks of mining complex data and depends strongly on the type of data that is input to the mining process. At this point, one could further extend the topic ontology by collecting a corpus of papers for a specific sub-domain, building a topic ontology and extending the appropriate node of the current DM topic ontology.

8.4.3 OntoDM domain coverage assessment

Since the data mining topic ontology contains only an IS-SUBTOPIC-OF taxonomy (using only IS-SUBTOPIC-OF relation) it is very difficult to perform direct comparison with the OntoDM ontology which does not include such a relation in its relation set. This is why we decided to perform the comparison at the level of data mining tasks, by examining the data mining task IS-A taxonomy of the OntoDM ontology and comparing it to the list of covered tasks within the topics identified in the topic ontology. We believe that this mapping of topics from the topic ontology to tasks in the manually constructed ones is well justified since the construction of the topic ontology was done by using abstracts and usually the data mining task addressed in the paper should be clearly stated in the abstract.

The OntoDM ontology contains an explicit taxonomy of data mining tasks, with the focus on predictive modeling tasks only (see Figure 6.11 in Chapter 6). The OntoDM task taxonomy has four basic data mining tasks at the top level. These includes tasks as: pattern discovery task, predictive modeling, clustering, and probability distribution estimation task. Further on the task taxonomy is developed based on the datatypes of the input data. This is extensively developed for the task of predictive modeling and more specifically tasks where the input data has a structured (complex) data type on the target side. The taxonomy allows the representation of tasks that work on data with arbitrary complex data types both on the descriptive side and the target side. This is very much in line with the topic ontology having a separate branch for complex data.

To summarize, this showed to be a positive line of reasoning and most of the identified topics from the topic ontology can be mapped to some of the OntoDM classes from the task taxonomy. However, there are some topics that are not represented as tasks in OntoDM. These include Inductive Logic Programming (ILP), reinforcement learning and the more extensive representation of mining of complex data. Some of the topics such as ILP, semi-supervised learning, data stream mining need to be more in detailed studied, to be directly represented in the OntoDM task taxonomy. Another topics, such as reinforcement learning is clearly out of scope for the OntoDM ontology. Regarding complex data, the OntoDT module, which provides representations of the datatypes, is still not developed deep enough to cover specific data from the application domains such as images, videos, and audio, so direct mapping from the images topic to the OntoDM tasks, for example, is not possible. The next step in the future work is to examine the articles present in the topic ontology, especially those not covered by the OntoDM and try to extract the most important data mining entities, and represent them adequately in the OntoDM ontology.

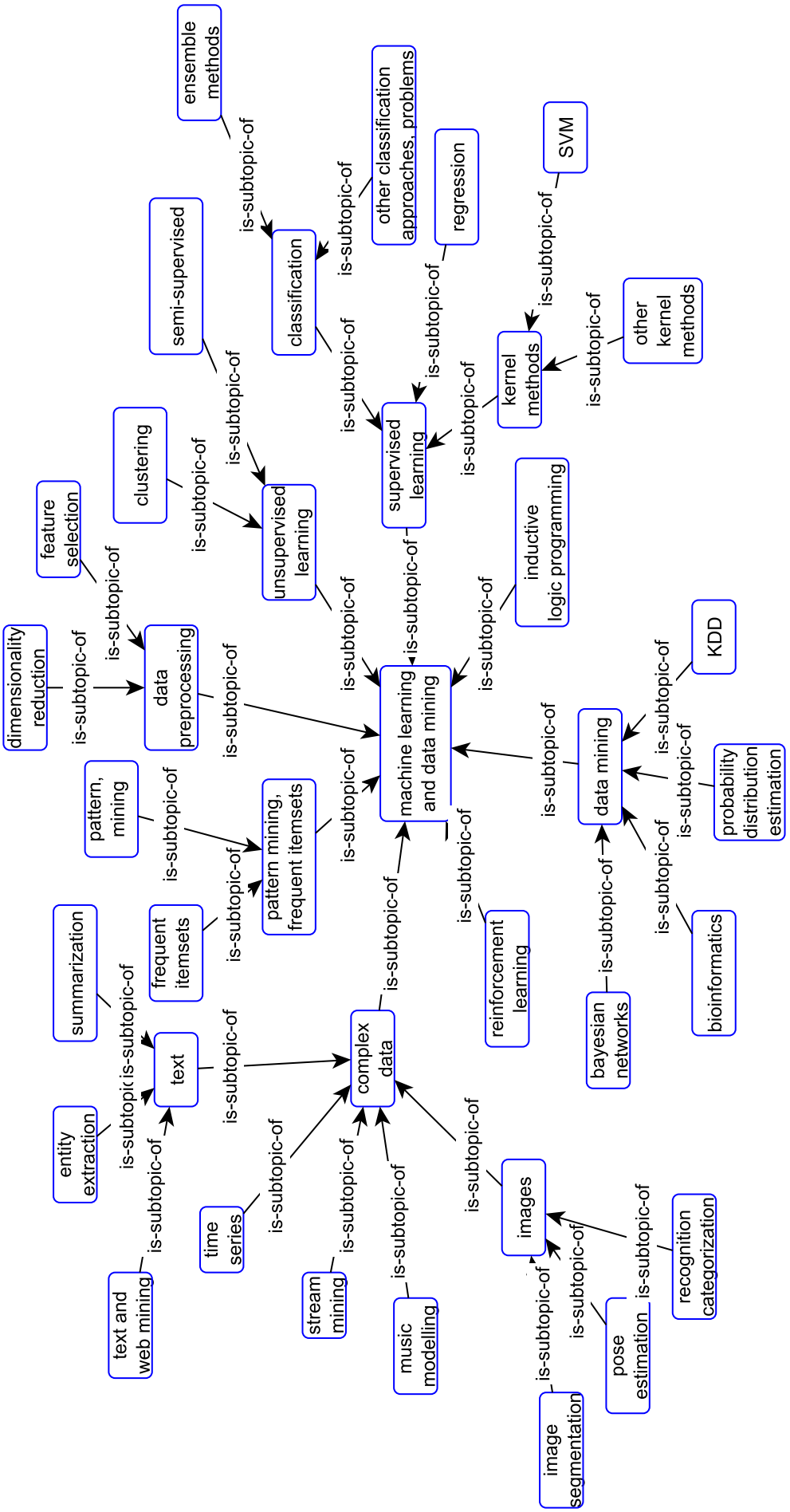


Figure 8.2: Data mining topic ontology obtained using domain input from a human expert.

9 Use cases

In previous chapters, we first presented the design and implementation of the OntoDM ontology (see Chapter 4), then we focused the analysis and discussion of all three OntoDM modules separately (see Chapter 5, 6, and 7), and finally presented the OntoDM evaluation (see Chapter 8). In this chapter, we present several use cases for the OntoDM ontology.

The OntoDM ontology can be used to facilitate different applications. These include applications in the context of automatization of the scientific method (representing and reasoning about data mining scenarios, workflows and experiments), semantic annotation of data mining algorithms and software, semantic annotation of articles describing data mining investigations, use of text mining with the ontology to harvest information about the used data mining methods from articles, etc. One of the unique features of OntoDM is that due to its design principles, it can be used in cross-domain applications, such as representing investigations within the automated drug discovery pipeline, .

In this section we present six use cases of the OntoDM ontology and its modules OntoDT, OntoDM-core, and OntoDM-KDD. First, we present how OntoDM can be used in the annotation of algorithms for the purpose of automatic reasoning (Section 9.1). Next, we present how OntoDM can be used for representing data mining scenarios (Section 9.2). Furthermore, we describe how OntoDM can be used for annotation of data mining investigations in application domains (Section 9.3). Next, we focus on demonstrating how OntoDM can be used to support cross-domain applications, as is the case for the Robot Scientist use case (Section 9.4). Finally, we conclude this chapter by first focusing on the use of OntoDM ontology as a mid-level ontology by other ontologies (Section 9.5) and the use of OntoDM together with text mining for annotation of data mining articles (Section 9.6).

9.1 Annotation of data mining algorithms: The Clus software

In this section, we present the use of the OntoDM ontology for annotation of algorithms for mining structured data. For that purpose, we discuss the example of annotating an algorithm from the Clus software system[1]. Clus is a decision tree and rule learning system that works in the predictive clustering framework (Blockeel et al., 1998). Along with the traditional data mining tasks, such as classification and regression task, the Clus system contains implementations of algorithms for mining structured data. This includes tasks such as single target and multi-target classification and regression (Blockeel et al., 1998; Zenko, 2007; Zenko and Dzeroski, 2008; Kocev et al., 2007; Aho et al., 2009), hierarchical classification (Vens et al., 2008; Schietgat et al., 2010), and time series prediction (Slavkov et al., 2010) (see Section 6.3.2 for definitions of these tasks). In addition, the Clus system includes an implementation of algorithm for constraint based induction of multi target regression trees (Struyf and Dzeroski, 2005), beam search induction of decision trees (Kocev et al., 2006), and an algorithm for the induction of decision trees with instance level constraints (Struyf and Dzeroski, 2007).

[1] Clus: `http://dtai.cs.kuleuven.be/clus/`

Table 9.1: Partial list of data mining algorithms implemented in the Clus software.

Algorithm	**Descriptive data specification**	**Output data specification**	**Data mining task**	**Generalization type**	**Publication**
Clus-STDT	tuple of primitives	boolean or discrete	flat classification task	classification tree	(Blockeel et al., 1998)
Clus-STRT	tuple of primitives	real	regression task	regression tree	(Blockeel et al., 1998)
Clus-MTDT	tuple of primitives	tuple of discrete or boolean	multi-target classification task	multi-target classification tree	(Blockeel et al., 1998)
Clus-MTRT	tuple of primitives	tuple of real	multi-target regression task	multi-target regression tree	(Blockeel et al., 1998)
Clus-STCR	tuple of primitives	boolean or discrete	flat classification task	regression rules	(Zenko, 2007)
Clus-STRR	tuple of primitives	real	regression task	regression rules	(Zenko, 2007)
Clus-MTCR	tuple of primitives	tuple of discrete or boolean	multi-target classification task	multi-target classification rules	(Zenko and Dzeroski, 2008)
Clus-MTRR	tuple of primitives	tuple of real	multi-target regression task	multi-target regression rules	(Zenko, 2007)
Clus-ensemble-STDT	tuple of primitives	boolean or discrete	flat classification task	ensemble of classification trees	(Kocev et al., 2007)
Clus-ensemble-STRT	tuple of primitives	real	regression task	ensemble of regression trees	(Kocev et al., 2007)
Clus-ensemble-MTDT	tuple of primitives	tuple of discrete or boolean	multi-target classification task	ensemble of regression trees	(Kocev et al., 2007)
Clus-ensemble-MTRT	tuple of primitives	tuple of real	multi-target regression task	ensemble of regression trees	(Kocev et al., 2007)
Clus-ensemble-MTRR	tuple of primitives	tuple of real	multi-target regression task	ensemble of regression rules	(Aho et al., 2009)
Clus-tree-HMC	tuple of primitives	tree with boolean edges and discrete nodes	tree-based hierarchical classification task	tree-based hierarchical classification tree	(Vens et al., 2008)
Clus-DAG-HMC	tuple of primitives	DAG with boolean edges and discrete nodes	DAG-based hierarchical classification task	DAG-based hierarchical classification tree	(Vens et al., 2008)
Clus-ensemble-tree-HMC	tuple of primitives	tree with boolean edges and discrete nodes	tree-based hierarchical classification task	ensemble of tree-based hierarchical classification trees	(Schietgat et al., 2010)
Clus-ensemble-DAG-HMC	tuple of primitives	DAG with boolean edges and discrete nodes	DAG-based hierarchical classification task	ensemble of DAG-based hierarchical classification trees	(Schietgat et al., 2010)
Clus-TS	tuple of primitives	sequence of reals	time-series prediction task	time-series classification tree	(Slavkov et al., 2010)
Clus-CBDTinduction	tuple of primitives	tuple of reals	constraint-based predictive modeling task	multi-target regression tree	(Struyf and Dzeroski, 2005)

In Table 9.1, we present a partial list of data mining algorithms implemented in the system. We present the descriptive and output data specification of the data the algorithm can be applied on, the data mining task that the algorithm solves and the generalization type it produces when executed on a dataset. For example, in the context of the OntoDM ontology, the Clus-ensemble-MTDT algorithm: (1) can be applied on data having tuple of primitives on the descriptive side, a tuple of booleans or discrete on the output side; (2) solves a multi-target classification task and (3) when executed on a dataset it produces as output an ensemble of multi-target classification trees.

In Section 6.3.4 of this thesis, we presented the OntoDM mechanism for representing knowledge about data mining algorithms for structured data. We can apply this methodology for the case of annotating data mining algorithms from the Clus software: (1) for the purpose of automatic reasoning over the knowledge about algorithms implemented in the software, and (2) to provide a schema for recording of data mining experiments performed by the execution of these algorithms in the context of experiment databases for mining structured data. In this use case, we present an example of how an algorithm from the Clus software can be annotated the terms from OntoDM.

In Figure 9.1, we present an example anotation of Clus-DAG-HMC algorithm from the Clus software, using terms from the OntoDM ontology. The `CLUS-DAG-HMC algorithm` is an instance of *DAG-based HC algorithm for multiple path and non-mandatory leaf node prediction using a global classifier approach* class (see Section 6.3.4.3). The algorithm solves the `DAG-based HC task for multiple path prediction and partial depth labeling` (see Section 6.3.2.3). In addition, the algorithm specification states the type of the generalization produced as output, in this case a `hierarchical classification tree specification` instance. It also it specifies the list of actions the algorithm performs when executed, represented by a `CLUS-DAG-HMC action specification` instance. Finally, it specifies the document where the algorithm has been published, represented by the `Vens et al.(2008) journal publication` instance.

The data mining task is the objective specification of the data mining algorithm at hand. The task is defined on data, characterized by a `descriptive tuple of primitives` as a descriptive data specification, and `output DAG with boolean edges and discrete nodes` as output data specification. For example, if we look at a specific dataset instance, e.g., `dataset ID0001`, it is characterized by a dataset specification instance denoting the type of the dataset e.g., `DAG-based HC dataset specification ID001` (see Section 6.3.1.2). This dataset specification is defined on data having the same data specifications as mentioned before.

`CLUS-DAG-HMC algorithm implementation` is an instance of *data mining algorithm implementation* class and is a concretization of the `CLUS-DAG-HMC algorithm`. An implementation has quality parameters (see Section 6.3.4.4). For example, `CLUS-DAG-HMC algorithm implementation` has as quality the `Wtype` parameter instance. This parameter defines how parents's class weights are aggregated in DAG shaped hierarchies of classes (see Vens et al. (2008) for more information).

The `CLUS-DAG-HMC module` is an instance of the *software module* class. It is an instance that is about the `CLUS-DAG-HMC algorithm implementation` and it has part `CLUS-DAG-HMC module description`. `The CLUS software` is an instance of the *software* class. It includes as its parts all of the CLUS modules, for example it includes the `CLUS-DAG-HMC module`. In addition, it includes, `JAVA programming language` instance as a specification of the programming language in which the software has been implemented, `version 2.12` instance as an information about software version, and `CLUS source code` instance as an information about the software source code location. Finally, `CLUS software` instance contains two relations with the organizations that produced the software (the `KULeuven` and `IJS` instances).

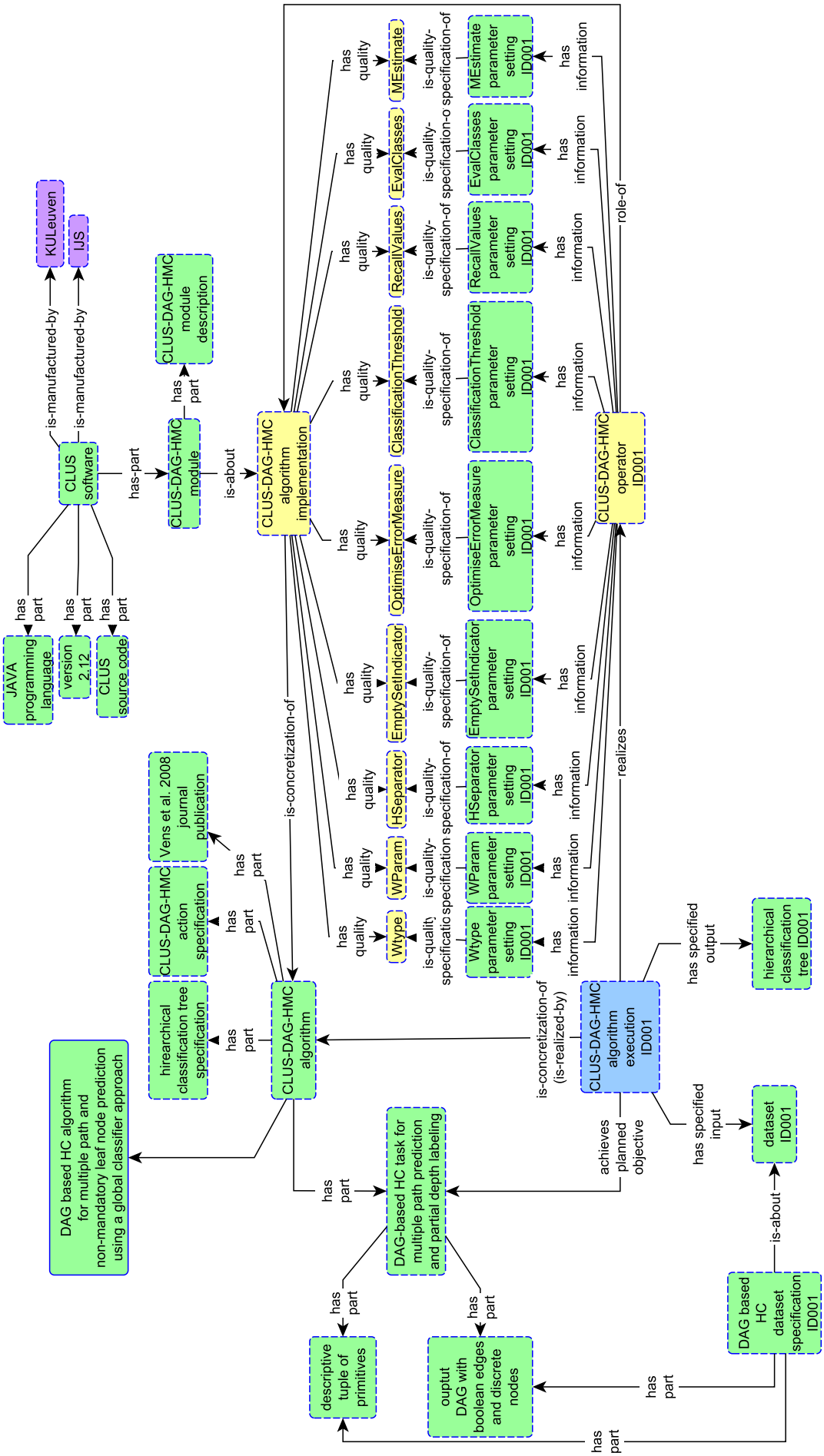


Figure 9.1: An example annotation of the Clus-DAG-HMC algorithm from the Clus software with OntoDM.

Along with the annotations of the algorithm specifications and implementation, we can use the OntoDM ontology to annotate the process of execution of an algorithm. This feature can be taken into account when designing experiment databases that contain also algorithms for mining structured data. For example, `CLUS-DAG-HMC algorithm execution ID001` is one instance denoting the process of execution of the `CLUS-DAG-HMC algorithm`. This instance has as specified input the `dataset ID001` instance and produces as output a the `hierarchical classification tree ID001` instance. In addition, the execution realizes a `CLUS-DAG-HMC operator ID001` that is a role of an algorithm implementation in the context of the execution process. The operator has information about the parameter setting of each parameter defined by the implementation for the specific execution process at hand. For example, `Wtype parameter setting ID0001` instance is an entity that is a quality specification of the `Wtype` parameter instance for the case of this specific algorithm execution process. In this way, we can use the OntoDM ontology to annotate all three levels of data mining algorithms: specifications, implementations and applications.

9.2 Representing data mining scenarios

In Section 6.3.6, we presented the OntoDM mechanism for representing data mining scenarios, data mining workflows and the process of executing data mining workflows. In this section, we present examples of use cases in the context of the use of the OntoDM ontology for representation and annotation of data mining scenarios used in data mining investigations. First, we present a representation of a simple and frequently used scenario for evaluation of predictive model (Section 9.2.1). Next, we present a representation of a more complex scenario for comparison of data mining algorithms (Section 9.2.2).

9.2.1 Scenario for evaluation of predictive models

In this section, we will focus on a representation of a data mining scenario for evaluating predictive models (see Figure 9.2). The predictive model evaluation, in general, includes the execution of a predictive modeling algorithm on a train set and obtaining a predictive model, executing a predictive model on a test set and obtaining a predicted set (which includes the predictions) and process of calculation of the evaluation using the predicted set and the test set. In the remainder of this subsection we present how this can be formally represented using the language of the OntoDM ontology.

A *predictive model evaluation scenario* is a sub-class of *data mining scenario*. The scenario is further concretized in a *predictive model evaluation workflow*. The workflow is realized in a *predictive model evaluation workflow execution*, which is a sub-class of *planned process*. A predictive model evaluation workflow is a complex entity and contains as parts three operators that are realized in the process of execution. As we already discussed in Section 6.3.4.4, an operator represents a role of an implementation of an algorithm in the context of executing it in a planned process. In this case, the workflow contains the following operators and/or realizable entities: *predictive modeling operator*, *predictive model*, and OntoDMpredictive model evaluation calculation operatorOntoDM.

Predictive modeling workflow execution is a complex process and consists of three sub-processes *predictive modeling algorithm execution*, *predictive model execution*, and *predictive model evaluation calculation*. The sub-processes are connected with a relation PRECEDED-BY, showing the ordering of the processes. The process achieves planned objective an *evaluation objective*. First, the *predictive modeling algorithm execution* process REALIZES a *predictive modeling operator*. It HAS-SPECIFIED-INPUT *dataset D1* and HAS-SPECIFIED-OUTPUT a *predictive model* (see Section 6.3.4.4). In this case, the *dataset D1* HAS-ROLE of a training set. Second, the *predictive model execution* process REALIZES a predictive model that is the output of the preceding process. In addition, this process HAS-SPECIFIED-INPUT *dataset D2*

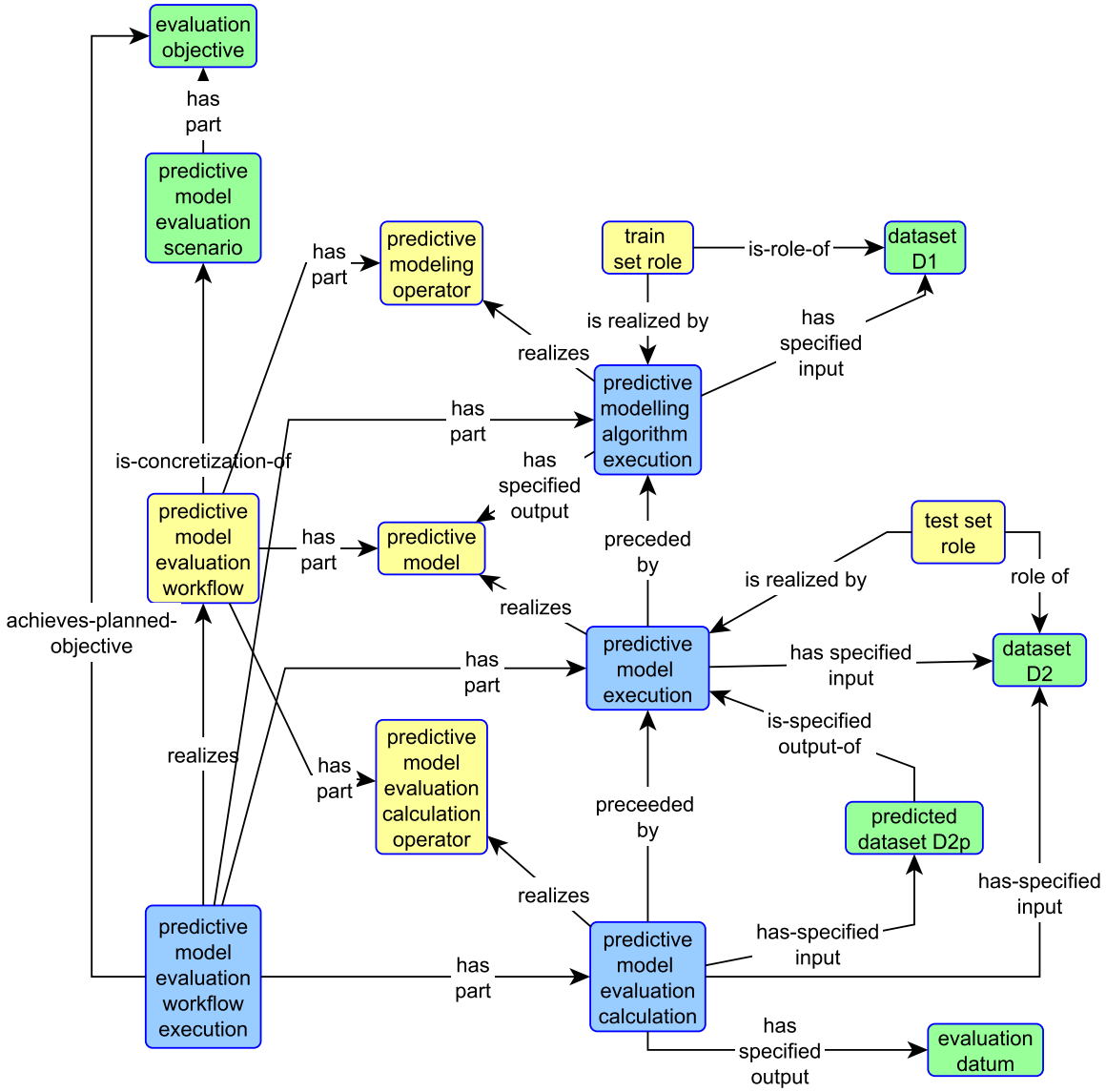


Figure 9.2: Predictive modeling evaluation scenario.

and it HAS-SPECIFIED-OUTPUT a *predicted dataset D2P* (see Section 6.3.3.3). In this case, the *dataset D2* HAS-ROLE of a test set.

Finally, a *predictive model evaluation calculation* is a *planned process*, that REALIZES a concretization of a *predictive modeling evaluation function specification*. The evaluation functions are dependent of the data mining task at hand. Examples of such functions are *root mean squared error* (for regression tasks) and *accuracy* (for classification tasks). The process has as inputs the *test set* and the *predicted set* and gives as output an *evaluation datum*. An *evaluation datum* IS-A *data item* that IS-SPECIFIED-OUTPUT-OF a *predictive modeling evaluation calculation* process and represents a value of the evaluation function.

9.2.2 Scenario for comparison of algorithms

OntoDM enables the formal representation of complex data mining investigations and scenarios. We will demonstrate this feature of OntoDM on an example DM scenario reported by Stojanova et al. (2010) in the paper "Estimating vegetation height and canopy cover from remotely sensed data with machine learning".

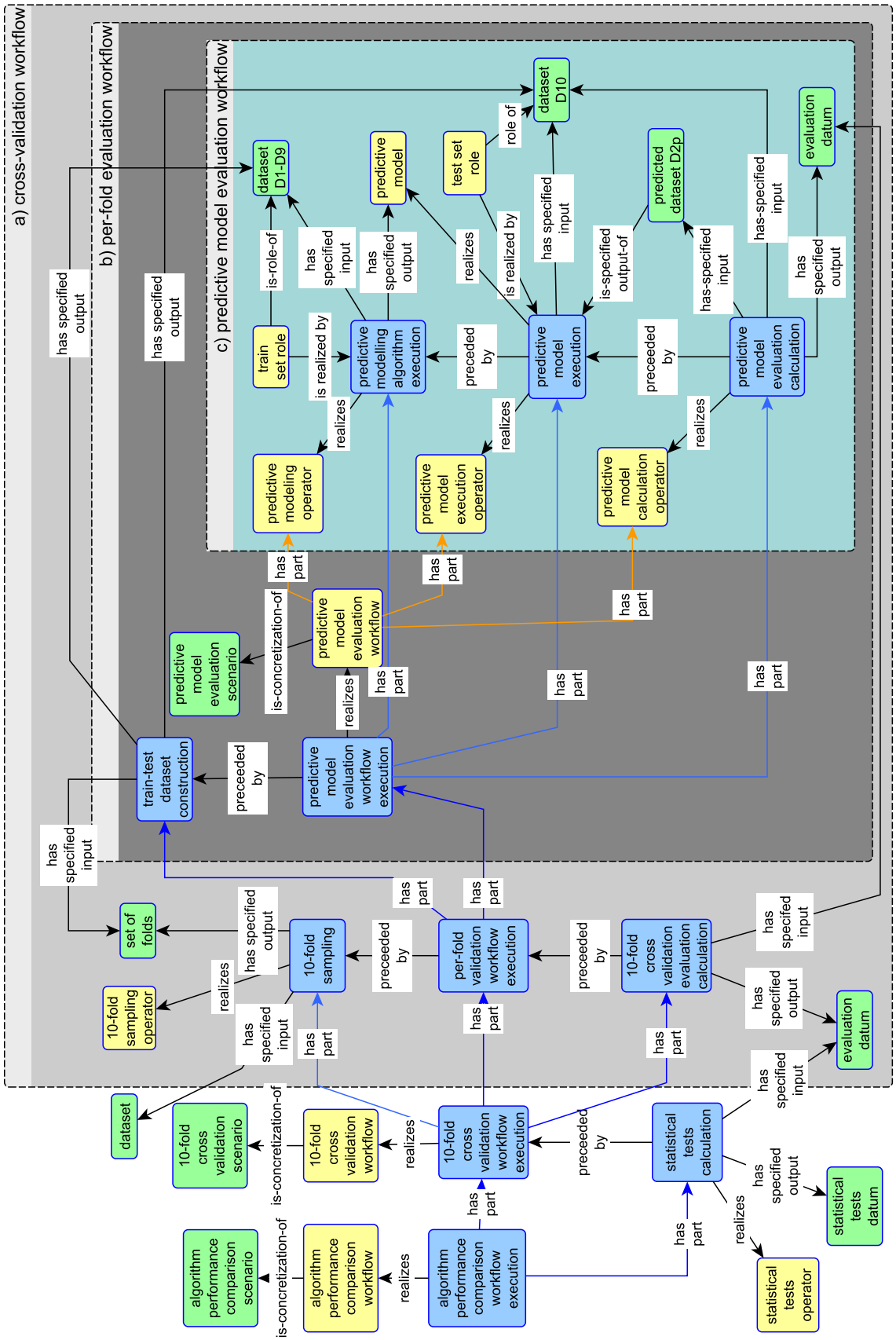


Figure 9.3: The OntoDM representation of an algorithm performance comparison scenario (a fragment).

Stojanova et al. (2010) use a set of data mining algorithms to model forest parameters. They compare the performance of different data mining algorithms for modeling forest parameters and discuss the use of the best model to generate forest vegetation maps. Such vegetation maps are of vital importance for forest management systems. In this work, the authors use a data mining scenario of the comparison of algorithms performance on the dataset at hand in order to decide which algorithm performs best on the dataset (see Section 3 from (Stojanova et al., 2010)). The OntoDM ontology allows the representation of such a scenario at different levels of granularity and flexibility (see Figure 9.3).

First, an *algorithm performance comparison scenario* is modeled in OntoDM as the subclass of the class *information entity* that contains basic information about the scenario, such as its title, author, description, objective, list of actions and the document where the scenario is published. The scenario is further concretisized as *algorithm performance comparison workflow*. A workflow contains operators and other embedded workflows as parts, and is further realized in the process of *workflow execution*.

The *algorithm performance comparison workflow execution* process has two sub-processes: *10-fold cross validation workflow execution* and *statistical tests calculation*. The *10-fold cross-validation workflow execution* is a realization of the 10-fold cross-validation workflow, while the *statistical tests calculation* is a realization of the *statistical test operator*.

The *10-fold cross-validation workflow execution* is a complex process and consists of three sub-processes (see Figure 9.3a): 10-fold sampling process, per-fold validation workflow execution, and 10-fold cross-validation evaluation calculation. The *10-fold sampling* process has as input the dataset and has as output a set of folds that is further used in the process of evaluation. The *per-fold validation workflow execution* performs predictive model evaluation per fold. The *cross validation calculation* process takes the output (evaluation datum) of each per-fold validation and calculates the evaluation datum for the complete cross-validation process. Finally, the cross-validation workflow is instantiated for each algorithm separately.

The *per-fold validation workflow execution* is one of the core processes in this scenario, and is instantiated for each of the folds (see Figure 9.3b). It is a complex process that involves the *train-test dataset construction*, using the set of folds, and the *predictive model evaluation workflow execution* (see Section 9.2.1). The latter (see Fig ure 9.3c) includes the processes of *predictive modeling algorithm execution*, where a predictive model is built by executing a DM algorithm on a training set, the process of *predictive model execution*, where the predictive model is executed on a test set, and *evaluation calculation*, where an evaluation datum is calculated for the desired evaluation function using as input the test set and the predicted dataset.

To summarize, the description of complex DM scenarios in natural language may be ambiguous. OntoDM allows unambiguous, semantically defined, and machine-processable representation of DM scenarios and workflows reported in literature.

9.3 Annotation of data mining investigations

In Chapter 7, we presented the OntoDM-KDD ontology module, which is a part of the OntoDM ontology that is dealing with the representation of data mining investigations modeled under the framework of the CRISP-DM methodology. In this section, we present an example of how the representation mechanism of OntoDM-KDD can be used to annotate data mining investigations in application domains of data mining. For this purpose, we focus on a data mining investigation titled "Estimating forest properties from remotely sensed data using data mining" summarized in a journal article by Stojanova et al. (2010), that was partially discussed in Section 9.2.2, where the focus was only on modeling the data mining scenario, which is just a small part of the complete investigation.

The data mining investigation described in the article by Stojanova et al. (2010) aims

at modeling forest properties, such as vegetation height and canopy cover, from remotely sensed data, by using data mining algorithms. The final goal of this investigation is to use the models of the properties to generate forest maps that can be deployed in forest management and forest decision support systems. The data mining investigation includes: the study of the application domain; a collection and generation of data for the process of modeling; preparation of the data for the process of modeling; modeling of the forest properties; evaluation of the modeling process and deciding on the best model; generation of the forest property maps; and finally deployment of the generated maps in forest management systems.

In Figure 9.4, we present a part of an annotation of a the data mining investigation summarized in Stojanova et al. (2010) journal article, using the OntoDM-KDD ontology module. First, we define `DM Investigation ID0001` as an instance of the *DM investigation* class. As we discussed in Section 7.3.1, a *data mining investigation* has as parts a planning process, a documenting process, and a KD phase execution process (or its child classes). For the case of our investigation, we define instances of these classes such as `planning ID001`, `documenting ID001`, and instances of the subclasses of *KD phase execution*. These include `application understanding ID001`, `data understanding ID001`, `data preparation ID001`, `modeling ID001`, `DM process evaluation ID001`, and `deployment ID001`. In addition, `'Estimating forest properties from remotely sensed data'` DENOTES the investigation and represents its title. Finally, we define the `investigation description ID001` instance that IS-ABOUT the investigation and represents its description.

The documentation process HAS-SPECIFIED-OUTPUT a *publication about an investigation*, which represents an entity that IS-ABOUT an investigation. `Publication about an investigation ID001` is an instance od this class and is used to represent the Stojanova et al. (2010) journal article. In addition, this instance has as parts document part instances (see Section 4.2.2.1), such as `abstract ID001`, `author list ID001`, `institution list ID001`, `introduction to a publication about an investigation ID001`, `methods section ID001`, `results section ID001`, `discussion section of a publication about an investigation ID001`, `conclusion to a publication about an investigation ID001`, and `references section ID001`. Finally, `'Estimating vegetation height and canopy cover from remotely sensed data with machine learning'` DENOTES the publication and represents its title.

The introduction part of the article, represented by the `introduction to a publication about an investigation ID001` instance, IS-ABOUT the application understanding process. It has as parts descriptions that are outputs of processes that compose the application understanding process. For example, the introduction includes description instances such as `application background description ID001`, `application objectives description ID001`, and `application success criteria description ID001`. These descriptions are instances of the *textual entity* class and are outputs of the `application objective identification ID0001` process instance. The same holds also for the other process instances that compose the application understanding process.

The methods section, represented by the `methods section ID001` instance, contains parts that are about the data understanding process, the data preparation process and modeling technique selection (as a sub-process of modeling). More specifically, it contains description of the study area (the Kras region), data sources (LiDAR and Landsat), data descriptions (descriptive and output variables), and description od the data mining techniques to be used. These descriptions are outputs of the the sub-processes of data understanding and data preparation, and the modeling technique selection process (for simplicity/readability reasons Figure 9.4 contains only the upper level processes).

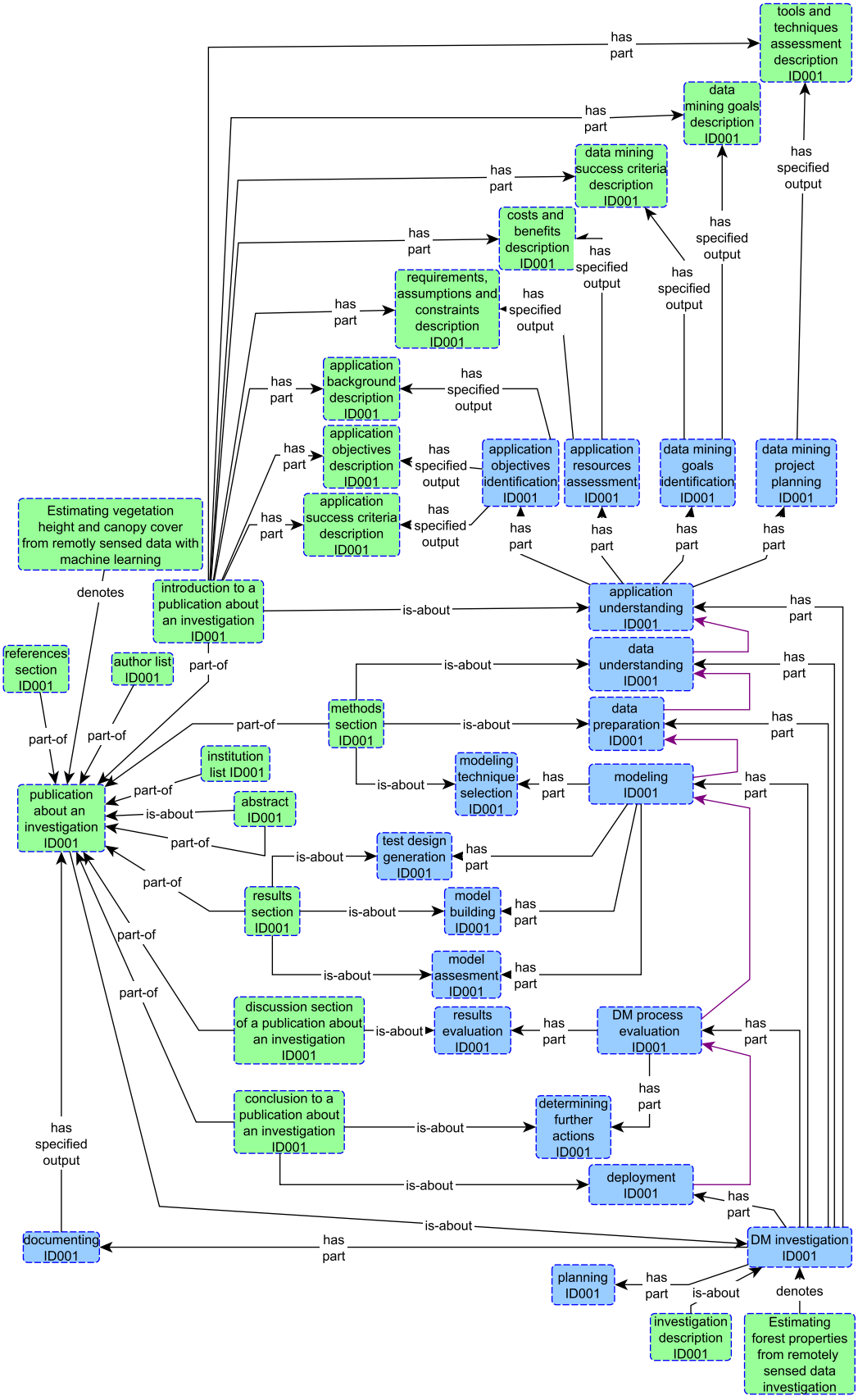


Figure 9.4: Part of an annotation of a data mining investigation summarized in a journal article with terms from the OntoDM ontology.

The results the section, represented by the `results section ID001` instance, contains parts that are about test design generation process, the model building process, and the model assessment process. These are all sub-processes of modeling. More specifically, it contains description of the experimental design, data mining algorithms applied, evaluation procedure and results (best models in terms of predictive performance and maps of vegetation height and canopy cover for the best model). These descriptions are outputs of the sub-processes of test design generation, model building and model assessment (for simplicity/readability reasons Fig. 9.4 contains only the upper level processes).

The discussion section, represented by the `discussion section of a publication about an investigation ID001` instance, contains parts that are about a results evaluation process, a sub-process of a DM process evaluation. More specifically, it contains descriptions of a comparison of the performances of all applied data mining techniques, a comparison to previous work, and a discussion of the produced maps of vegetation parameters.

The conclusion section, represented by the `conclusion to a publication about an investigation ID001` instance, contains parts that are about the deployment process and determining further actions process, a sub-process of DM process evaluation. More specifically, it contains a summary of contributions, a description of the deployment of the produced maps, and a description of envisioned future work.

9.4 The Robot Scientist use case: An ontology-based representation of QSAR modeling for drug discovery

Quantitative Structure Activity Relationship (QSAR) modeling is one of the key components of the drug discovery pipeline. A QSAR modeling algorithm is usually a DM algorithm. It receives as input a description of compounds with associated pharmacological activities and outputs a predictive model of activity. The modeling process provides a mapping from the structure of compounds to their activity.

The current representation of modeling and evaluation of industrial QSARs (such as OpenQSAR[2]) does not allow the recording of all necessary information for the evaluation of predictive models. Industrial QSARs are also limited to feature-based labeled datasets with primitive output. Typically, compound descriptors are used as features on the descriptive side and the activity of the compound on the selected biological target on the output side.

The OntoDM ontology was designed to support, among others, the representation of the QSAR modeling process for the Robot Scientist project[3]. A Robot Scientist "Eve" is an automated laboratory designed to carry out autonomous drug discovery investigations. Eve offers an intelligent and economical approach for lead compound discovery. It is able to switch from mass screening to QSAR automatically controlled by active machine learning techniques. Eve avoids the use of a large compound library and the screening of all available, often costly, chemical compounds.

Eve can detect the pattern in compound structures during mass screening, compare it with background knowledge, determine what compounds are most likely to have high activity, and experiment further with such compounds. Eve works not only with the conventional representation of chemical compounds as a table of features, but also with an innovative relational representation of chemical structure (King et al., 1996). One of the goals of the Robot Scientist project is to compare the effectiveness of various data mining algorithms, including inductive and active learning, on different representations of datasets.

The DDI ontology (for Drug Discovery Investigations) has been developed to support the recording of data and metadata generated by Eve in a formally defined semantic form

[2]OpenQSAR: `www.openqsar.com/`

[3]Robot Scientist Project: `www.aber.ac.uk/en/cs/research/cb/projects/robotscientist/`

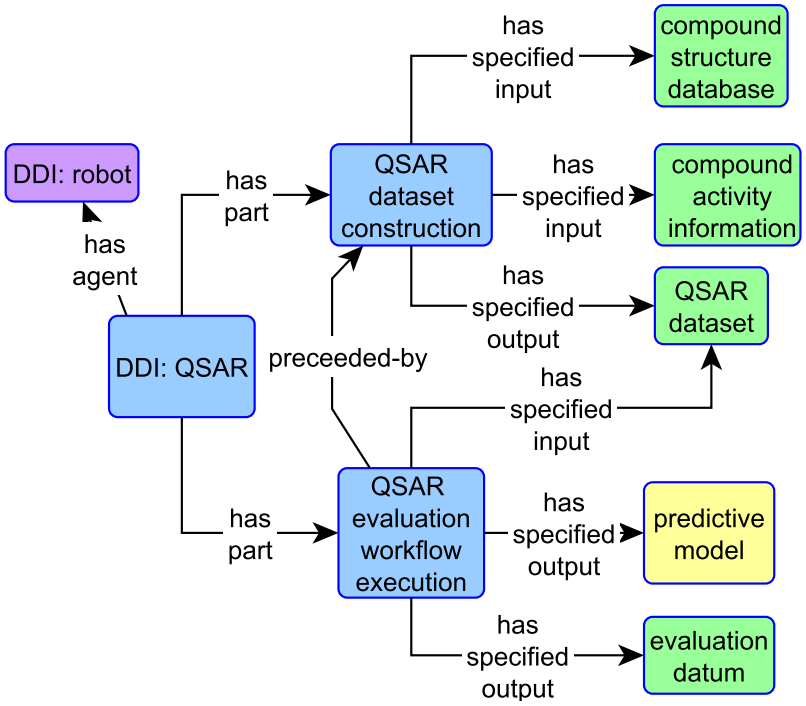


Figure 9.5: QSAR modeling for drug discovery investigations with OntoDM and DDI.

(Qi et al., 2010). OntoDM and DDI were built upon the same design principles, share the same set of relations, and are based on the same upper level classes, and therefore are fully interoperable. These ontologies can be downloaded together, i.e., into Protege, and used to represent the principal entities for the recording of QSAR modeling. The OntoDM ontology enables accurate recording of all key aspects of the use of data mining for QSAR modeling. It provides logically defined semantic representations of datasets, complex datatypes (propositional and graph-like), QSAR algorithms, predictive modeling tasks, and QSAR model evaluations within the context of a drug discovery investigation.

In Figure 9.5, we present an upper-level overview of the representation of the QSAR process using both OntoDM (unmarked boxes) and DDI (boxes with the mark DDI:). The class *QSAR* process is modeled in DDI as a part of the class *investigation* process, and HAS-AGENT a *robot*. The QSAR process has two sub-processes: *QSAR dataset construction* and *QSAR evaluation workflow execution* process.

The *QSAR dataset construction* process is performed by an autonomous agent - the robot scientist Eve. The robot collects input data about compounds from a *compound structure database* and *compound activity information* from the mass screening assays. The output of this process is a *QSAR dataset*. A *QSAR dataset* IS-A *dataset* and it inherits all the properties of the OntoDM *dataset* class. The power of the OntoDM representation of structured data allows accurate recording of molecular structures by structured datatypes.

The process of *QSAR evaluation workflow execution* is a complex process that involves building of a QSAR model on training data, executing the model on test data and calculation of evaluation functions. OntoDM allows the precise recording of the evaluation process and its specifications (e.g., parameter settings of operators). Finally, OntoDM can be further used for automated selection of QSAR algorithms based on selected tasks and types of generalizations (linear models, decision rules, regression trees, neural networks) and available datasets.

9.5 OntoDM-core as a mid-level ontology: the Expose ontology

There is a pressing need for storing machine learning experiments and their results in public databases. Currently, thousands of machine learning research papers contain extensive experimental results, but details of those experiments are often lost after publication. This makes it impossible to reproduce, reuse, or compare such experiments. Vanschoren et al. (2012) have addressed this need by the implementation of a machine learning experiment database that at present holds over 650,000 experiments: "Experiments are automatically transcribed in a common language" and "are uploaded to pre-designed databases where they are stored in an organized fashion: the results of every experiment are linked to the exact underlying components (such as algorithm, parameter settings and dataset used)". This common language (ExpML), the database design and an overall framework are based on the ontology Exposé (Vanschoren et al., 2012). Exposé, in turn, uses OntoDM as a mid-level ontology (see Figure 9.6).

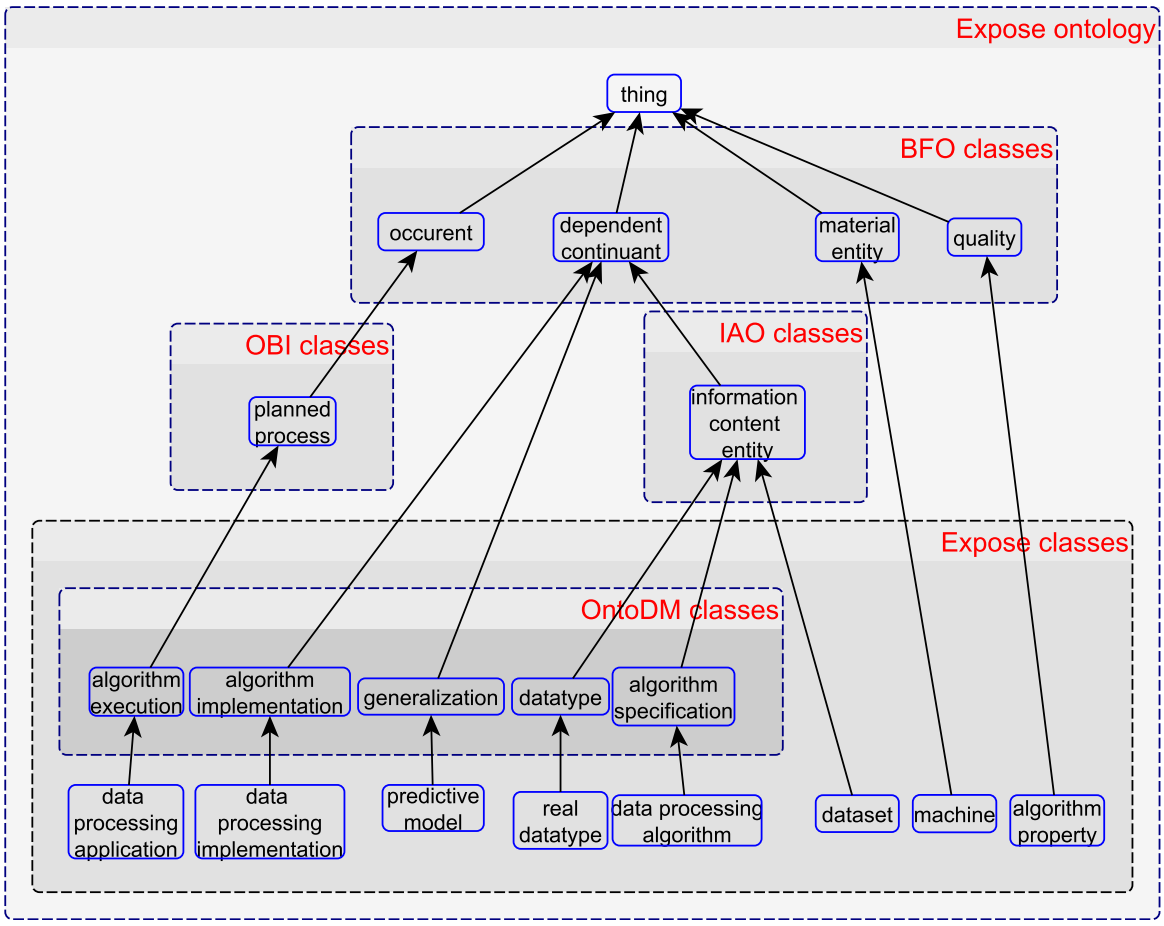


Figure 9.6: The reuse of OntoDM classes by Expose.

Exposé and OntoDM follow the same design principles and therefore are fully interoperable. Exposé reuses the OntoDM three representational levels (specification, implementation, and application). Moreover, it reuses and further extends the classes related to algorithms: *algorithm specification*, *algorithm implementation* and *algorithm execution* (or application) for the tasks of supervised classification and regression. In addition, it reuses the OntoDM classes representing primitive datatypes, constraints, predictive models, mathematical functions, and evaluation measures.

The machine learning experiments database not only collects all the details on the ex-

periments, performed and shared by many researchers, but also enables a wide range of new interesting research queries for future extensive studies. In the future, we envision the extension of Exposé (and experiment databases) to cover the description of experiments for mining structured data: This will involve reusing the OntoDM mechanism and taxonomies for representing structured datasets, DM tasks, generalizations, and DM algorithms.

9.6 The text mining use case: annotation of DM articles

Ontologies are widely used in text mining for a number of tasks, e.g., named entity recognition (NER), disambiguation of terms, event extraction, and others. OntoDM, as a generic ontology for the scientific domain of data mining, can assist in text mining of DM articles. It can do so by providing the terminology to describe DM investigations.

We will demonstrate the usability of OntoDM for text mining on an example paper by Ford et al. (2004) from the ART Corpus of Biochemistry papers[4]. The CheTA (Chemistry using Text Annotations) project by the UK National Centre for Text Mining[5] used the Open-Source Chemistry Analysis Routines (OSCAR) toolkit for NER in chemistry publications to annotate papers from the ART Corpus. OSCAR employs domain ontologies to support annotation and three OBO ontologies, ChEBI, FIX, and REX, were used in the CheTA project. However, depending on the goals of annotation, i.e., identification of most popular methods for prediction of biological activity of compounds and extraction of information about such methods, or reasoning about molecular descriptors and chemical diversity, it may be desirable to use other domain specific ontologies (such as OntoDM).

The considered paper by Ford et al. reports on QSAR studies with a new molecular descriptor EVA. OntoDM could assist in NER of QSAR modeling terms, such as *training/test set*, *regression model*, *cross-validation*, while the OBO ontologies would fail to do so. The paper has three instances of the class *cross-validation*: Leave-One-Out Cross-Validation, Leave-n-Out Cross-Validation, and Leave-Group-Out Cross-Validation. It also has two instances of the class *regression model*: PLS regression model and QSAR regression model;

The paper further concernes three instances of the class *dataset*: the Calcium Channel Agonist data set, 36 compound data set, and proprietary data sets (Shell Research Ltd., the former Sittingbourne Research Centre). It also has several statements about *training/test set*: "test set of 76 compounds" and "the training set is comprised of the same number of well-defined components". OntoDM could also assist in the recognition of other DM relevant terms, i.e., "robustness of the model" is a *generalization quality* and "model validity" is a *generalization evaluation*. OntoDM may be used for reasoning over recognized entities through the defined relations between those entities.

Furthermore, OntoDM may assist in the disambiguation of terms. For example, the term EVA was classified by OSCAR as a chemical compound with a likelihood 0.22. OntoDM, on the other hand, recognizes "biological activity EVA" as an instance of the class *feature* via the synonym "molecular descriptor".

[4]Art corpus: `www.aber.ac.uk/en/cs/research/cb/projects/art/art-corpus/`
[5]NACTEM centre: `www.nactem.ac.uk/cheta/`

10 Conclusions

In this thesis, we propose a reference modular ontology for the domain of data mining OntoDM. The ontology is composed of three modules covering different aspects of data mining - OntoDT, OntoDM-core, and OntoDM-KDD. The OntoDT module supports the representation of knowledge about datatypes and is based on an accepted ISO standard for datatypes in computer systems. The OntoDM-core module formalizes the key data mining entities needs for representing the mining of structured data in the context of a general framework for data mining. The OntoDM-KDD module formalizes the knowledge discovery process, supports the representation of data mining investigations: It is based on the CRISP-DM process model, where CRISP-DM stands for Cross Industry Standard Process for Data Mining.

The OntoDM ontology is designed and implemented by following ontology best practices and design principles. It includes an upper-level ontology BFO as a template, uses formally defined relations from RO and other state-of-the-art ontologies, and reuses classes and relations from other ontologies for representing scientific investigations, such as OBI, IAO, EXACT, and SWO, having them as mid-level ontologies. In addition, we proposed a three-level description structure (specification, implementation, application) for representing entities from the domain of data mining, where each description level represents classes with fundamentally different nature. Finally, the ontology is developed in a general fashion in order to be used as a mid-level ontology and can be easily extended further by other ontologies that focus on a specific part of the data mining domain.

The OntoDM ontology supports a large variety of applications, as demonstrated by the use cases presented. OntoDM supports the annotation and representation of data mining algorithms, data mining scenarios, and data mining investigations. Furthermore, it provides support for representing the Quantitative structure-activity relationship (QSAR) modeling process in the context of Drug Design Investigations. Finally, the OntoDM ontology can be used as a mid-level ontology and combined with text mining to support the annotation of articles containing data mining terms.

10.1 Contributions: The OntoDM ontology of data mining

The major contribution of this thesis is the OntoDM ontology. The work presented in this thesis comprises several contributions to the area of data mining, knowledge discovery in databases, and applied ontology. After performing a State-of-the-Art literature survey on ontologies for representing scientific investigations and ontologies in the domain of data mining, we made the following contributions:

1. *We designed and implemented a modular reference domain ontology of data mining OntoDM, composed of three modules OntoDT, OntoDM-core, and OntoDM-KDD.* Due to the use of the upper-level ontology BFO and the extensive reuse of classes from the OBI and the IAO mid-level ontologies, we proposed a three-level description structure for representing scientific investigations that includes the specification level (representing information entities), the implementation level (representing realizable entities),

and the application level (representing processual entities). Each description level represents classes with fundamentally different nature explicitly shown with a sub-class relation to an upper-level category, and consequently covers different aspects of the domain of interest. The ontology is implemented in the OWL-DL language, developed with the Protege tool, and available online at `http://www.ontodm.com`.

(a) *We designed and implemented an ontology module for representing knowledge about datatypes OntoDT, based on the ISO 11404 standard for general purpose datatypes.* The ontology module contains:

- Representation of datatypes, datatype qualities, and datatype characterizing operations.
- Taxonomy of datatypes, datatype qualities and characterising operations.
- Generic mechanism for representing arbitrarily complex datatypes.

(b) *We designed and implemented an ontology module for representing knowledge about the core data mining entities for representing mining structured data OntoDM-core, in a context of a general framework for data mining.* This module contains:

- Representation of datasets, dataset specifications, and a taxonomy of datasets.
- Representation of data mining tasks, a taxonomy of fundamental data mining tasks, a taxonomy of predictive modeling tasks, and a taxonomy of hierarchical classification tasks.
- Representation of generalizations in a three-level description structure, a taxonomy of generalizations, and a taxonomy of predictive models.
- Representation of data mining algorithms in a three-level description structure, a taxonomy of data mining algorithms, a taxonomy of predictive modeling algorithms, and a taxonomy of hierarchical classification algorithms.
- Generic structure for representing algorithms in computer science.
- Representation of constraints, constraint-based data mining tasks, a taxonomy of constraints, and a taxonomy of constraint based data mining tasks.
- Representation of data mining scenarios in a three-level description structure.

(c) *We designed and implemented an ontology module for representing knowledge about data mining investigations OntoDM-KDD, based on the CRISP-DM methodology.* This module contains:

- Representation of data mining investigations in a three-level representation structure.
- Representation of the phases in a data mining investigation (such as application understanding, data understanding, data preparation, modeling, DM process evaluation, and deployment) in a three-level representation structure.
- Representation of the inputs and outputs of the different phases of a data mining investigation, such as descriptions and reports.

We preformed evaluation of the OntoDM ontology and its modules OntoDT, OntoDM-core and OntoDM-KDD. First, we assessed the ontology by discussing the values of the ontology metrics calculated for the ontology and its modules. Next, the evaluation was performed by assessing the ontology towards a set of design principles and best practices, and assessing whether the competency questions be formalized in the language of the ontology. In addition, we provided a domain coverage assessment by comparing with a data mining topic ontology.

2. *We demonstrated the use of the OntoDM ontology on six different use cases.* These include the:

   (a) Use of OntoDM for annotating of data mining algorithms.

   (b) Use of OntoDM for representing of data mining scenarios.

   (c) Use of OntoDM for annotating of data mining investigations.

   (d) Use of OntoDM along with the DDI (Drug Design Investigation) ontology for supporting ontology-based representation of QSAR modeling in drug discovery within the context of The Robot Scientist project.

   (e) Use of OntoDM by other users as a mid-level ontology for representing experiments in machine learning.

   (f) Use of OntoDM to support the annotation of data mining articles using text mining.

The novelties that the OntoDM ontology introduces and what distinguishes it from other related ontologies are the facts that it allows representation of mining of structured data and the general process of data mining in a principled way, it is based on a theoretical ontological framework and due to this it can be connected to other domain ontologies to support cross-domain applications. The OntoDM ontology is also the first ontology that supports the representation of the complete process of knowledge discovery.

10.2 Advantages of OntoDM over existing data mining ontologies

In this section, we present a critical comparison of the OntoDM ontology, proposed in this thesis, over other developed ontologies (Bernstein et al., 2005; Žáková et al., 2010; Diamantini and Potena, 2008; Kietz et al., 2009; Cannataro and Comito, 2003; Brezany et al., 2007; Hilario et al., 2009; Vanschoren et al., 2012), described in more detail in Section 3.2.

If we first analyze the *ontological aspects* of these ontologies, almost all existing ontologies of data mining are not explicitly aligned to any upper-level ontology and do not use a set of formal community agreed (or logically defined) relations. This leads to the non-compatibility of ontologies and to ontologies that can not be easily reused and extended. An exception is the Expose ontology (Vanschoren et al., 2012), which is based on the same design principles as OntoDM and even uses it as a mid-level ontology. In addition, no assessment has been performed for any of the existing ontologies of DM.

The strength and the advantages of the OntoDM ontology in this sense is that it is based on a theoretical ontological framework (that includes upper-level ontology, formal relations, and extensive reuse of already developed resources) that allows easy extensions and reuse as demonstrated in Section 9.5, This allows an easy connection to other domain ontologies to support cross domain applications as demonstrated in Section 9.4. Finally, OntoDM has been thoroughly evaluated using principles from ontology engineering.

Regarding the *domain coverage* aspect, other proposed ontologies deal mostly with representation of mining propositional, single table, data with primitive attributes as datatypes and focusing on predictive data mining tasks, such as classification and regression. An exception is the KD Ontology (Žáková et al., 2010) that includes some very specific relational mining tasks and algorithms. None of existing ontologies of DM provide a flexible mechanism for extension and generalization toward representing data mining tasks and algorithms for mining structured data.

The strength and advantage of the OntoDM ontology in this sense is that it is developed in a general fashion to provide the representation of mining of structured data (e.g.,

structured output prediction) and constraint based data mining. It is easily extendable to cover new data mining tasks and algorithms that operate on data from arbitrarily complex datatypes (provided by the OntoDT module). The weakness of the OntoDM ontology in this phase of development is that it does not contain large number of instances. For this purpose other developed ontologies can be reused (such as DMOP and Expose), especially for the single table data mining tasks and algorithms.

In the context of *representation of the complete knowledge discovery process*, most of the ontologies focus only on representing the modeling phase. Some of the ontologies, such as DMOP (Hilario et al., 2009) and Expose (Vanschoren et al., 2012), also provide enities that cover the data preparation phase of a KDD process, but do not provide ontological support for the complete KDD process. The strength of the OntoDM ontology in this context is that it provides support for representation of the complete knowledge discovery process (i.e., data mining investigations) from the application understanding phase to the final deployment phase; through the OntoDM-KDD module.

10.3 Public availability of OntoDM

The OntoDM ontology, presented in this thesis, was developed as an open source ontology. We have created a dedicated web page `http://www.ontodm.com` where we provide up to date versions of the ontology and support information about the ontology, such as publications. In addition, to promote the ontology within the application domains (mostly biological and biomedical) of data mining, the ontology is also available from the BioPortal[1]. The BioPortal offers the possibility of obtaining ontology metrics and provides excellent visualization capabilities.

10.4 Future work

In future developments of the OntoDM ontology, outlined in this thesis, we plan to focus on several aspects. First, we plan to align and map our ontology to other upper-level ontologies such as YAMATO (Mizoguchi, 2010) and GFO (Herre et al., 2006). For example, a mapping to YAMATO would additionally provide OntoDM with the possibility of representing change (e.g., in the case of dynamic datasets), more expressiveness for representing events, and a well developed framework for dealing with representations. Second, the design of an ontology of data mining, that focuses on information, and not physical entities (as is the case in the biomedical domains) reveals the need of organizing and establishing a separate ontology of relations between information entities. Such an ontology would import relations from the current State-of-the-Art ontologies such as OBI, IAO, SWO, and RO relational ontology, and would modify their definitions to directly support information entities (e.g., modify and extend the HAS-PART relation with HAS-INFORMATION-PART). We have already started developments along these lines (Soldatova et al., 2012). Next, the relation ontology for informational entities would be extended with specific relations required by the domain of application. For example, the OntoDM ontology would require the HAS-MEMBER relation, which is the relation between a set and its members, to specifically denote that a dataset HAS-MEMBER data examples. Finally, we will provide an alignment of OntoDM to the newly proposed ontology of relations for information entities.

In the context of data mining entities, we plan to extend the established ontological framework for representing entities about components of data mining algorithms, such as distance functions and kernel functions. This would enable the formulation of a taxonomy of distance and kernel functions based on the information about the type of data. Next, we want to populate the ontology downward with instances. In that context, having in mind the

[1]BioPortal: `http://bioportal.bioontology.org/`

generality of the OntoDM ontology with respect to different datatypes, and treating single-table data mining as a special case, we can import terms from other ontologies representing data mining entities: Several such ontologies are being actively developed and treat in detail different aspects of single-table data mining, e.g., DMOP (Hilario et al., 2009) and Exposé (Vanschoren and Soldatova, 2010). Importing terms from the DMOP ontology directly is more difficult, since the ontology has a larger set of incompatible relations that are not based on upper-level classes, so ontology mapping would need to be performed to establish a relationship between the two ontologies. Importing classes from the Exposé ontology is more straightforward as this ontology is built on the same principles as OntoDM and even uses some of the OntoDM classes. Finally, we want to use the OntoDM framework to support the representations of dynamical systems, their behavior and predictive models such as systems of ordinary differential equations.

In the context of applications, we plan to employ OntoDM ontology to further support cross-domain applications, as is the case with The Robot Scientist application. We also want to develop in more detail the application for annotating data mining investigations, with the ultimate goal of providing a tool for this purpose having as a backbone the OntoDM ontology. We further want to extend the experiments database framework proposed by Blockeel (2006) and implemented by Blockeel and Vanschoren (2007) (for the case of classification and regression), for the general case of representing experiments for mining structured data, again having as backbone the OntoDM ontology. Finally, having built the ontology in a modular fashion every ontology module can be used independently as a separate ontology. In particular, we expect OntoDT to have a variety of applications in different application areas: Extensions of OntoDT towards better descriptions of datatypes used in bioinformatics would thus be derived.

We would like to include more contributors from the domain of data mining into the development of OntoDM and the discussion about how to represent data mining entities. Hopefully, this would lead to the development of standards for the domain, as is the case in biomedical domains. In November 2010, the OntoDM, DMOP and Exposé ontology projects have formed the DMO (Data Mining Ontologies) Foundry with the aim of developing the DMO Core ontology which will standardize the formal definitions of principal data mining entities such as algorithm, implementation, application, task, dataset, datatype, etc. Finally, applying the OntoDM design principles to the development of ontologies for other areas of computer science, is one of the most important long term objectives of our research.

11 Acknowledgements

Many people have supported me and my research, either directly or indirectly, in the last couple of years, and this thesis would not exist without their support. First, I would like to thank my supervisor Prof. Dr. Sašo Džeroski for giving me the opportunity to work in the area of data mining and for his constant support on the work on this thesis. His ideas and suggestions were the basis for the work presented in this thesis.

I would also like to thank the members of my PhD committee for providing comments the thesis and providing useful feedback. These comments improved the quality of the thesis and to place my work in a broader context.

My work and research was directly supported by several funding bodies. I wish to thank all of them for the financial support all these years. These include the European Commission, through the research project IQ - Inductive Queries for Mining Patterns and Models, where I invested lot of my energy and work in the first three years of my stay in Slovenia, and the Slovenian Research Agency, through the local projects.

Many thanks go also to Dr. Larisa Soldatova who introduced me to the field of ontology and ontology engineering. She was always extremely motivating, whenever we had a chance to collaborate either on some article or in general ontology discussions. I also enjoyed my three visits to the Aberystwyth University in Wales, where we had an opportunity to work more closer on the ontological topics, but also had periods of the day where we would just talk about non-research related topics and have fun.

The colleagues at the Department of Knowledge Technologies are responsible for the pleasant work environment. I would like to thank all of you, but especially the heart of our department Mili Bauer, our department secretary, who was always willing to listen to whatever problems I had and assist in solving them as much as she could. Also, I would like to thank my office mates Bernard Ženko, Martin Žnidarsič, and Elena Ikonomovska for indirectly helping me to survive the PhD writing process in the office.

One of my passions in life is opera, and everybody who knows me good understands my deep connection with this art. In this sense I would like to thank all of my operatic friends throughout the world, who helped me indirectly in the tedious research process, to free my mind in the endless discussions about opera productions, singers, and composers. Big thanks to Benjamin Virc for joining me on all the operatic adventures in the regional opera theaters in the last year, also the huge help with the translation of the abstract of the thesis in Slovene.

I would like also to thank endlessly to my Ljubljana family: Dragi Kocev, Ivica Slavkov, and Aleksandra Raškovska. Without your everyday support this thesis would not become alive.

I would like to thank my close family, my mother Maja, father Petar, sister Bruna, and my grandmother Vinka for supporting me in all steps in life, without your unselfish love this thesis would not exist. At the end, I dedicate this thesis to the memory of my grandfather Bruno, as he was the reason I decided to pursue my education to the highest level and I think he would have been proud of me.

12 References

Agrawal, R.; Imielinski, T.; Swami, A. N. Mining association rules between sets of items in large databases. In: Buneman, P.; Jajodia, S. (eds.) *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data.* 207–216 (ACM Press, 1993).

Aho, T.; Ženko, B.; Džeroski, S. Rule ensembles for multi-target regression. In: Wang, W.; Kargupta, H.; Ranka, S.; Yu, P. S.; Wu, X. (eds.) *ICDM 2009, The Ninth IEEE International Conference on Data Mining.* 21–30 (IEEE Computer Society, 2009).

Aristotle; Warrington, J. *Aristotle's metaphysics* (Dent; Dutton, London, New York, 1956).

Arpírez, J. C.; Corcho, O.; Fernández-López, M.; Gómez-Pérez, A. Webode in a nutshell. *AI Magazine* **24**, 37–47 (2003).

Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. A.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin, G. M.; Sherlock, G. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature Genetics* **25**, 25–29 (2000).

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; Patel-Schneider, P. F. (eds.). *The Description Logic Handbook: Theory, Implementation, and Applications* (Cambridge University Press, 2003).

Bakir, G. H.; Hofmann, T.; Schölkopf, B.; Smola, A. J.; Taskar, B.; Vishwanathan, S. V. N. (eds.) *Predicting Structured Data. Neural Information Processing* (The MIT Press, 2007).

Ball, C. A.; Brazma, A. MGED standards: work in progress. *OMICS: A Journal of Integrative Biology* **10**, 138–144 (2006).

Bayardo, R. (ed.) *Constraints in Data Mining*, *SIGKDD Explorations* **4** (2002).

Bernaras, A.; Laresgoiti, I.; Corera, J. Building and reusing ontologies for electrical network applications. In: Wahlster, W. (ed.) *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI'96).* 298–302 (John Wiley and Sons, Chichester, UK, 1996).

Bernstein, A.; Provost, F.; Hill, S. Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering* **17**, 503–518 (2005).

Blockeel, H. Experiment databases: A novel methodology for experimental research. In: *Knowledge Discovery in Inductive Databases, 4th International Workshop, KDID'05, Revised, Selected and Invited Papers*, *Lecture Notes in Computer Science* **3933**, 72–85 (Springer, 2006).

Blockeel, H.; Raedt, L. D.; Ramon, J. Top-down induction of clustering trees. In: *Proceedings of the 15th International Conference on Machine Learning.* 55–63 (Morgan Kaufmann, 1998).

Blockeel, H.; Vanschoren, J. Experiment databases: Towards an improved experimental methodology in machine learning. In: *Knowledge Discovery in Databases: PKDD 2007*, *Lecture Notes in Computer Science* **4702**, 6–17 (Springer, 2007).

Bozsak, E.; Ehrig, M.; Handschuh, S.; Hotho, A.; Maedche, A.; Motik, B.; Oberle, D.; Schmitz, C.; Staab, S.; Stojanovic, L.; Stojanovic, N.; Studer, R.; Stumme, G.; Sure, Y.; Tane, J.; Volz, R.; Zacharias, V. Kaon - towards a large scale semantic web. In: Bauknecht, K.; Tjoa, A. M.; Quirchmayr, G. (eds.) *E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Proceedings*, *Lecture Notes in Computer Science* **2455**, 304–313 (Springer, Berlin, 2002).

Brachman, R.; Levesque, H. *Knowledge Representation and Reasoning* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004).

Brank, J.; Grobelnik, M.; Mladenic, D. A survey of ontology evaluation techniques. In: *Zbornik 8. mednarodne multikonference Informacijska družba IS 2005*. 166–169 (2005).

Brazma, A.; Hingamp, P.; Quackenbush, J.; Sherlock, G.; Spellman, P.; Stoeckert, C.; Aach, J.; Ansorge, W.; Ball, C. A.; Causton, H. C.; Gaasterland, T.; Glenisson, P.; Holstege, F. C.; Kim, I. F.; Markowitz, V.; Matese, J. C.; Parkinson, H.; Robinson, A.; Sarkans, U.; Schulze-Kremer, S.; Stewart, J.; Taylor, R.; Vilo, J.; Vingron, M. Minimum information about a microarray experiment (MIAME)-toward standards for microarray data. *Nature Genetics* **29**, 365–371 (2001).

Brezany, P.; Janciak, I.; Tjoa, A. M. Ontology-based construction of grid data mining workflows. In: Nigro, H. O.; Cisaro, S. E. G.; Xodo, D. H. (eds.) *Data Mining with Ontologies: Implementations, Findings and Frameworks*. 182–210 (IGI Global, 2007).

Brinkman, R. R.; Courtot, M.; Derom, D.; Fostel, J. M.; He, Y.; Lord, P.; Malone, J.; Parkinson, H.; Peters, B.; Rocca-Serra, P.; Ruttenberg, A.; Sansone, S.-A. A.; Soldatova, L. N.; Stoeckert, C. J.; Turner, J. A.; Zheng, J.; OBI consortium. Modeling biomedical experimental processes with OBI. *Journal of biomedical semantics* **1**(Suppl 1), S7 (2010).

Buitelaar, P.; Cimiano, P. (eds.) *Ontology learning and population: bridging the gap between text and knowledge* (IOS Press, 2008).

Cannataro, M.; Comito, C. A data mining ontology for GRID programming. In: *Proc. 1st Intl. Wshp. on Semantics in Peer-to-Peer and Grid Computing*. 113–134 (2003).

Cannataro, M.; Talia, D. The knowledge GRID. *Communications of the ACM* **46**, 89–93 (2003).

Caruana, R. Multitask learning. *Machine Learning* **28**, 41–75 (1997).

Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; Wirth, R. *CRISP-DM 1.0 Step-by-step data mining guide* (The CRISP-DM consortium, 2000).

Chapman, P.; Kerber, R.; Clinton, J.; Khabaza, T.; Reinartz, T.; Wirth, R. The CRISP-DM process model. *Discussion Paper* (1999).

Chebotko, A.; Deng, Y.; Lu, S.; Fotouhi, F.; Aristar, A.; Brugman, H.; Klassmann, A.; Sloetjes, H.; Russel, A.; Wittenburg, P. Ontoelan. an ontology-based linguistic multimedia annotator. In: *Proceedings of the IEEE Sixth International Symposium on Multimedia Software Engineering*, ISMSE '04. 329–336 (IEEE Computer Society, Washington, DC, USA, 2004).

Cook, D. J.; Holder, L. B. *Mining Graph Data* (John Wiley & Sons, 2006).

Cortes, C.; Vapnik, V. Support-vector networks. *Machine Learning* **20**, 273–297 (1995).

Courtot, M.; Gibson, F.; Lister, A. L.; Malone, J.; Schober, D.; Brinkman, R. R.; Ruttenberg, A. MIREOT: The minimum information to reference an external ontology term. *Applied Ontology* **6**, 23–33 (2011).

de Matos, P.; Alcántara, R.; Dekker, A.; Ennis, M.; Hastings, J.; Haug, K.; Spiteri, I.; Turner, S.; Steinbeck, C. Chemical entities of biological interest: an update. *Nucleic Acids Research* **38**(suppl 1), D249–D254 (2010).

Demšar, D.; Džeroski, S.; Larsen, T.; Struyf, J.; Axelsen, J.; Bruns-Pedersen, M.; Krogh, P. H. Using multi-objective classification to model communities of soil. *Ecological Modelling* **191**, 131–143 (2006).

Diamantini, C.; Potena, D. Semantic annotation and services for KDD tools sharing and reuse. In: *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*. 761–770 (IEEE Computer Society, 2008).

Diamantini, C.; Potena, D.; Storti, E. KDDONTO: An ontology for discovery and composition of kdd algorithms. In: Podpečan, V.; Lavrač, N.; Kok, J.; de Bruin, J. (eds.) *Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*. 13–25 (2009).

Diamantini, C.; Potena, D.; Storti, E. Supporting users in kdd process design: a semantic similarity matching approach. In: *Proc. of the 3rd Planning To Learn Workshop at ECAI 2010s*. 27–34 (2010).

Dietterich, T.; Domingos, P.; Getoor, L.; Muggleton, S.; Tadepalli, P. Structured machine learning: the next ten years. *Machine Learning* **73**, 3–23 (2008).

Dong, G. *Sequence Data Mining* (Springer-Verlag, Berlin, Heidelberg, 2009).

Džeroski, S. Towards a general framework for data mining. In: *Knowledge Discovery in Inductive Databases 5th Intl. Wshp., KDID 2006 Revised Selected and Invited Papers*, *Lecture Notes in Computer Science* **4747**, 259–300 (Springer, 2007).

Džeroski, S.; Langley, P.; Todorovski, L. Computational discovery of scientific knowledge. In: Džeroski, S.; Todorovski, L. (eds.) *Computational Discovery of Scientific Knowledge*, *Lecture Notes in Computer Science* **4660**, 1–14 (Springer, 2007).

Farquhar, A.; Fikes, R.; Rice, J. The ontolingua server: a tool for collaborative ontology construction. *International Journal of Human Computer Studies* **46**, 707–727 (1997).

Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. From data mining to knowledge discovery: An overview. In: Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*. 1–34 (AAAI, 1996).

Feldman, R.; Sanger, J. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data* (Cambridge University Press, 2006).

Ford, M.; Phillips, L.; Ste, A. Optimising the eva descriptor for prediction of biological activity. *Organic & Biomolecular Chemistry* **2**, 3301–3311 (2004).

Fortuna, B.; Grobelnik, M.; Mladenic, D. Ontogen: semi-automatic ontology editor. In: *Proceedings of the 2007 conference on Human interface: Part II*. 309–318 (Springer, 2007).

Fox, M. S.; Grüninger, M. Ontologies for enterprise integration. In: *CoopIS*, 82–89 (1994).

Garcia, J.; García-Penalvo, F. J.; Theron, R. A survey on ontology metrics. In: Lytras, M. D.; Ordonez De Pablos, P.; Ziderman, A.; Roulstone, A.; Maurer, H.; Imber, J. B. (eds.) *Knowledge Management, Information Systems, E-Learning, and Sustainability Research*, *Communications in Computer and Information Science* **111**, 22–27 (Springer, 2010).

Genesereth, M.; Nilsson, N. *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann, 1987).

Gómez-Pérez, A. Ontology evaluation. In: Staab, S.; Studer, R. (eds.) *Handbook on Ontologies*. 251–274 (Springer, 2004).

Grenon, P.; Smith, B. Snap and span: Towards dynamic spatial ontology. *Spatial Cognition & Computation* **4**, 69–104 (2004).

Grobelnik, M.; Mladenić, D. Knowledge Discovery for Ontology Construction. In: Davies, J.; Studer, R,; Warren, P. (eds.) *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. 9–27 (John Wiley & Sons, Ltd, 2006).

Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**, 199–220 (1993).

Grüninger, M.; Fox, M. Methodology for the Design and Evaluation of Ontologies. In: *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing*. 1–10 (1995).

Guarino, N. *Formal Ontology in Information Systems: Proceedings of the 1st International Conference* (IOS Press, 1998).

Guarino, N.; Giaretta, P. Ontologies and Knowledge Bases: Towards a Terminological Clarification. *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*. 25–32 (1995).

Han, J.; Kamber, M. *Data Mining: Concepts and Techniques* (Morgan Kaufmann Publishers, 2005).

Hand, D. J.; Smyth, P.; Mannila, H. *Principles of data mining* (MIT Press, 2001).

Hardy, B.; Douglas, N.; Helma, C.; Rautenberg, M.; Jeliazkova, N.; Jeliazkov, V.; Nikolova, I.; Benigni, R.; Tcheremenskaia, O.; Kramer, S.; Girschick, T.; Buchwald, F.; Wicker, J.; Karwath, A.; Gutlein, M.; Maunz, A.; Sarimveis, H.; Melagraki, G.; Afantitis, A.; Sopasakis, P.; Gallagher, D.; Poroikov, V.; Filimonov, D.; Zakharov, A.; Lagunin, A.; Gloriozova, T.; Novikov, S.; Skvortsova, N.; Druzhilovsky, D.; Chawla, S.; Ghosh, I.; Ray, S.; Patel, H.; Escher, S. Collaborative development of predictive toxicology applications. *Journal of Cheminformatics* **2**, 7 (2010).

Herre, H.; Heller, B.; Burek, P.; Hoehndorf, R.; Loebe, F.; Michalek, H. *General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles.* (University of Leipzig, 2006).

Hilario, M.; Kalousis, A.; Nguyen, P.; Woznica, A. A data mining ontology for algorithm selection and Meta-Mining. In: Podpečan, V.; Lavrač, N.; Kok, J.; de Bruin, J. (eds.) *Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*. 76–88 (2009).

Hilario, M.; Nguyen, P.; Do, H.; Woznica, A.; Kalousis, A. Ontology-based meta-mining of knowledge discovery workflows. In: Jankowski, N.; Duch, W.; Grabczewski, K. (eds.) *Meta-Learning in Computational Intelligence*, *Studies in Computational Intelligence* **358**, 273–315 (Springer, 2011).

Hornick, M. F.; Marcadé, E.; Venkayala, S. *Java Data Mining: Strategy, Standard, and Practice: A Practical Guide for architecture, design, and implementation* (Morgan Kaufmann, 2006).

International Organization for Standardization. *ISO/IEC 11404:1996 - Information technology – Programming languages, their environments and system software interfaces – Language-independent datatypes* (International Organization for Standardization, 1996). `http://www.iso.org/iso/catalogue_detail.htm?csnumber=19346`.

International Organization for Standardization. *ISO/IEC 11404:2007 - Information technology – General-Purpose Datatypes (GPD)* (International Organization for Standardization, 2007). `http://www.iso.org/iso/catalogue_detail.htm?csnumber=39479`.

Kalousis, A.; Bernstein, A.; Hilario, M. Meta-learning with kernels and similarity functions for planning of data mining workflows. In: Brazdil, P.; Bernstein, A.; Hunter, L. (eds.) *Proceedings of the Second Planning to Learn Workshop (PlanLearn) at the ICML/COLT/UAI*. 23–28 (2008).

Karalic, A.; Bratko, I. First order regression. *Machine Learning* **26**, 147–176 (1997).

Kaufman, L.; Rousseeuw, P. *Finding Groups in Data An Introduction to Cluster Analysis* (Wiley Interscience, 1990).

Kerrigan, M. Wsmoviz: An ontology visualization approach for wsmo. In: *Proceedings of the conference on Information Visualization*. 411–416 (IEEE Computer Society, 2006).

Kietz, J.; Serban, F.; Bernstein, A.; Fischer, S. Towards cooperative planning of data mining workflows. In: Podpečan, V.; Lavrač, N.; Kok, J.; de Bruin, J. (eds.) *Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*. 1–13 (2009).

Kietz, J.-U.; F. Serban, A. B.; Fischer, S. Data mining workflow templates for intelligent discovery assistance and Auto-Experimentation. In: *ECML/PKDD 2010 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-10)*. 1–12 (2010).

Kifer, M.; Lausen, G.; Wu, J. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* **42**, 741–843 (1995).

King, R.; Rowland, J.; Oliver, S. G.; Young, M.; Aubrey, W.; Byrne, E.; Liakata, M.; Markham, M.; Pir, P.; Soldatova, L. N.; Sparkes, A.; Whelan, K.; Clare, A. The Automation of Science. *Science* **324**, 85–89 (2009).

King, R. D.; Muggleton, S. H.; Srinivasan, A.; Sternberg, M. J. Structure-activity relationships derived by machine learning: the use of atoms and their bond connectivities to predict mutagenicity by inductive logic programming. *Proceedings of the National Academy of Sciences* **93**, 438–442 (1996).

Kocev, D.; Celine, V.; Struyf, J.; Džeroski, S. Ensembles of multi-objective decision trees. In: *Machine Learning: ECML 2007, 18th European Conference on Machine Learning, Proceedings*, *Lecture Notes in Computer Science* **4701**, 624–631 (Springer, 2007).

Kocev, D.; Džeroski, S.; White, M.; Newell, G.; Griffioen, P. Using single and multi-target regression trees and ensembles to model a compound index of vegetation condition. *Ecological Modeling* **220**, 1159–1168 (2009).

Kocev, D.; Struyf, J.; Dzeroski, S. Beam search induction and similarity constraints for predictive clustering trees. In: Dzeroski, S.; Struyf, J. (eds.) *Knowledge Discovery in Inductive Databases*, *Lecture Notes in Computer Science* **4747**, 134–151 (Springer, 2006).

Kononenko, I.; Kukar, M. *Machine Learning and Data Mining: Introduction to Principles and Algorithms* (Horwood Publishing Limited, 2007).

Kriegel, H.-P.; Borgwardt, K.; Kröger, P.; Pryakhin, A.; Schubert, M.; Zimek, A. Future trends in data mining. *Data Mining and Knowledge Discovery* **15**, 87–97 (2007).

Lamprecht, A.-L.; Naujokat, S.; Steffen, B.; Margaria, T. Constraint-guided workflow composition based on the EDAM ontology. In: *Proceedings of the 3rd International Workshop on Semantic Web Applications and Tools for the Life Sciences* (2010).

Lavrač, N.; Novak, P. K.; Mozetic, I.; Podpečan, V.; Motaln, H.; Petek, M.; Gruden, K. Semantic subgroup discovery: Using ontologies in microarray data analysis. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* **1**, 5613–5616 (IEEE, 2009).

Lenat, D. B. Cyc: a large-scale investment in knowledge infrastructure. *Communications of the ACM* **38**, 33–38 (1995).

Lister, A.; Lord, P.; Pocock, M.; Wipat, A. Annotation of SMBL models through rule-based semantic integration. *Proceedings of the Bio-Ontologies Special Interest Group Meeting 2009: Knowledge in Biology* (2009).

López, M. F.; Gómez-Pérez, A.; Sierra, J. P.; Sierra, A. P. Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems* **14**, 37–46 (1999).

Maccagnan, A.; Riva, M.; Feltrin, E.; Simionati, B.; Vardanega, T.; Valle, G.; Cannata, N. Combining ontologies and workflows to design formal protocols for biological laboratories. *Automated Experimentation* **2**, 3 (2010).

MacQueen, J. B. Some methods for classification and analysis of multivariate observations. In: Cam, L. M. L.; Neyman, J. (eds.) *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* **1**, 281–297 (University of California Press, 1967).

Malaia, E. *Engineering ontology: domain acquisition methodology and practice* (VDM Verlag Dr. Muller, Saarbrucken, 2009).

Mannila, H.; Toivonen, H. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* **1**, 241–258 (1997).

Masolo, C.; Borgo, S.; Gangemi, A.; Guarino, N.; Oltramari, A. *Ontology Library - final*. (Laboratory For Applied Ontology - ISTC-CNR - Trento, Italy, 2003).

Meek, B. A taxonomy of datatypes. *SIGPLAN Notes* **29**, 159–167 (1994).

Melli, G. Concept mentions within KDD-2009 abstracts (KDD09cma1) linked to a kdd ontology (kddo1). In: Calzolari, N.; Choukri, K.; Maegaard, B.; Mariani, J.; Odijk, J.; Piperidis, S.; Rosner, M.; Tapias, D. (eds.) *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC'10)*. 3817–3820 (2010).

Mizoguchi, R. Tutorial on ontological engineering: Part 1: Introduction to ontological engineering. *New Generation Computing* **21**, 365–384 (2003).

Mizoguchi, R. Yamato: Yet another more advanced top-level ontology (2010). http://www.ei.sanken.osaka-u.ac.jp/hozo/onto_library/YAMATO101216.pdf.

Mizoguchi, R.; Vanwelkenhuysen, J.; Ikeda, M. Task ontology for reuse of problem solving knowledge. In: *Knowledge Building and Knowledge Sharing 1995 at the 2nd International Conference on Very Large-Scale Knowledge Bases* **4275**, 46–59 (Enschede, The Netherlands, 1995).

Morik, K.; Scholz, M. The miningmart approach to knowledge discovery in databases. In: Zhong, N.; Liu, J.;(eds), *Intelligent Technologies for Information Analysis*. 47–65 (Springer, 2004).

Munn, K.; Smith, B. *Applied Ontology: An Introduction*, *Metaphysical research* **9** (Ontos Verlag, 2008).

Neches, R.; Fikes, R.; Finin, T.; Gruber, T.; Patil, R.; Senator, T.; Swartout, W. R. Enabling technology for knowledge sharing. *AI Magazine* **12**, 36–56 (1991).

Niles, I.; Pease, A. Towards a standard upper ontology. In: *Proceedings of the international conference on Formal Ontology in Information Systems (FOIS '01)*. 2–9 (ACM, 2001).

Noy, N.; Fergerson, R.; Musen, M. The knowledge model of Protege-2000: Combining interoperability and flexibility. *Lecture Notes in Computer Science* **1937**, 69–82 (2000).

Obrst, L. Ontological architectures. In: Poli, R.; Healy, M.; Kameas, A. (eds.) *Theory and Applications of Ontology: Computer Applications*. 27–66 (Springer Netherlands, 2010).

Obrst, L.; Ceusters, W.; Mani, I.; Ray, S.; Smith, B. The evaluation of ontologies. In: Baker, C. J. O.; Cheung, K.-H. (eds.) *Semantic Web*. 139–158 (Springer, 2007).

Panov, P.; Džeroski, S.; Soldatova, L. N. OntoDM: An ontology of data mining. In: *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*. 752–760 (IEEE Computer Society, 2008).

Panov, P.; Soldatova, L. N.; Džeroski, S. Towards an ontology of data mining investigations. In: *Discovery Science, 12th International Conference, DS 2009*, *Lecture Notes in Computer Science* **5808**, 257–271 (Springer, 2009).

Panov, P.; Soldatova, L.; Džeroski, S. Representing entities in the OntoDM data mining ontology. In: Džeroski, S.; Goethals, B.; Panov, P. (eds.) *Inductive Databases and Constraint-Based Data Mining*. 27–58 (Springer New York, 2010).

Panov, P.; Soldatova, L.; Džeroski, S. Generic Ontology of Data Mining. *Journal of Web Semantics*, (2012). (under review)

Peng, Y.; Kou, G.; Shi, Y.; Chen, Z. A descriptive framework for the field of data mining and knowledge discovery. *International Journal of Information Technology and Decision Making* **7**, 639–682 (2008).

Pettifer, S.; Ison, J.; Kalaš, M.; Thorne, D.; McDermott, P.; Jonassen, I.; Liaquat, A.; Fernández, J. M.; Rodriguez, J. M.; Partners, I.; Pisano, D. G.; Blanchet, C.; Uludag, M.; Rice, P.; Bartaseviciute, E.; Rapacki, K.; Hekkelman, M.; Sand, O.; Stockinger, H.; Clegg, A. B.; Bongcam-Rudloff, E.; Salzemann, J.; Breton, V.; Attwood, T. K.; Cameron, G.; Vriend, G. The embrace web service collection. *Nucleic Acids Research* **38**(suppl 2), W683–W688 (2010).

Podpečan, V.; Zemenova, M.; Lavrač, N. Orange4WS environment for service-oriented data mining. *The Computer Journal* (2011).

Porzel, R.; Malaka, R. A task-based approach for ontology evaluation. In: *Proc. of ECAI 2004 Workshop on Ontology Learning and Population* (2004).

Qi, D.; King, R.; Hopkins, A.; Bickerton, G. R.; Soldatova, L. An ontology for description of drug discovery investigations. *Journal of Integrative Bioinformatics* **7**, 126 (2010).

Ramakrishnan, R.; Agrawal, R.; Freytag, J.-C.; Bollinger, T.; Clifton, C. W.; Dzeroski, S.; Hipp, J.; Keim, D.; Kramer, S.; Kriegel, H.-P.; Leser, U.; Liu, B.; Mannila, H.; Meo, R.; Morishita, S.; Ng, R.; Pei, J.; Raghavan, P.; Spiliopoulou, M.; Srivastava, J.; Torra, V. Data mining: The next generation. In: Agrawal, R.; Freytag, J. C.; Ramakrishnan, R. (eds.) *Perspectives Workshop: Data Mining: The Next Generation*, *Dagstuhl Seminar Proceedings* (Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Dagstuhl, Germany, 2005).

Rosse, C.; Mejino, J. L. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics* **36**, 478–500 (2003).

Schietgat, L.; Vens, C.; Struyf, J.; Blockeel, H.; Kocev, D.; Dzeroski, S. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC Bioinformatics* **11**, 2 (2010).

Silla, C.; Freitas, A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* **22**, 31–72 (2011).

Simperl, E. P. B.; Tempich, C. Ontology engineering: A reality check. In: Meersman, R.; Tari, Z. (eds.) *OTM Conferences (1)*, *Lecture Notes in Computer Science* **4275**, 836–854 (Springer, 2006).

Sirin, E.; Parsia, B. SPARQL-DL: SPARQL query for OWL-DL. In: *3rd OWL Experiences and Directions Workshop (OWLED-2007)* (2007).

Slavkov, I.; Gjorgjioski, V.; Struyf, J.; Džeroski, S. Finding explained groups of time-course gene expression profiles with predictive clustering trees. *Molecular BioSystems* **6**, 729–740 (2010).

Smith, B. *Blackwell Guide to the Philosophy of Computing and Information*, chap. *Ontology* (Oxford Blackwell, 2003a).

Smith, B. Ontology. In: Floridi, L. (ed.) *Blackwell Guide to the Philosophy of Computing and Information*. 155–166 (Oxford Blackwell, 2003b).

Smith, B.; Ashburner, M.; Rosse, C.; Bard, J.; Bug, W.; Ceusters, W.; Goldberg, L. J.; Eilbeck, K.; Ireland, A.; Mungall, C. J.; Leontis, N.; Rocca-Serra, P.; Ruttenberg, A.; Sansone, S.-A.; Scheuermann, R. H.; Shah, N.; Whetzel, P. L.; Lewis, S. The OBO foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology* **25**, 1251–1255 (2007).

Smith, B.; Ceusters, W.; Klagges, B.; Kohler, J.; Kumar, A.; Lomax, J.; Mungall, C.; Neuhaus, F.; Rector, A. L.; Rosse, C. Relations in biomedical ontologies. *Genome Biology* **6**, R46 (2005).

Smith, B.; Kusnierczyk, W.; Schober, D.; Ceusters, W. Towards a reference terminology for ontology research and development in the biomedical domain. In: *Proceedings of the Second International Workshop on Formal Biomedical Knowledge Representation: "Biomedical Ontology in Action" (KR-MED 2006)*. 57–65 (2006).

Smith, B.; Shah, N. Ontologies for biomedicine - how to make them and use them. (Tutorial notes at ISMB/ECCB, 2007).

Soldatova, L. N.; Aubrey, W.; King, R. D.; Clare, A. The EXACT description of biomedical protocols. *Bioinformatics* **24**, i295–303 (2008).

Soldatova, L. N.; King, R. D. Are the current ontologies in biology good ontologies? *Nature Biotechnology* **23**, 1095–1098 (2005).

Soldatova, L.; Džeroski, S.; Panov, P. Relations for information entities. In: *Proceedings of Bio-Ontologies SIG, co-located with ISMB 2012* (2012) (to appear)

Soldatova, L. N.; King, R. D. An ontology of scientific experiments. *Journal of the Royal Society Interface* **3**, 795–803 (2006).

Sowa, J. F. *Knowledge representation: logical, philosophical and computational foundations* (Brooks/Cole Publishing Co., 2000).

Staab, S.; Studer, R. (eds.) *Handbook on Ontologies. International Handbooks on Information Systems* (Springer, 2004).

Ste, R. D.; Baker, P. G.; Bechhofer, S.; Ng, G.; Jacoby, A.; Paton, N.; Goble, C. A.; Brass, A. Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics* **16**, 184–186 (2000).

Stojanova, D.; Panov, P.; Gjorgjioski, V.; Kobler, A.; Dzeroski, S. Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics* **5**, 256–266 (2010).

Struyf, J.; Dzeroski, S. Constraint based induction of multi-objective regression trees. In: Bonchi, F.; Boulicaut, J.-F. (eds.) *Knowledge Discovery in Inductive Databases*, *Lecture Notes in Computer Science* **3933**, 222–233 (Springer, 2005).

Struyf, J.; Džeroski, S. Clustering trees with instance level constraints. In: Kok, J. N.; Koronacki, J.; de Mántaras, R. L.; Matwin, S.; Mladenic, D.; Skowron, A. (eds.) *European Conference on Machine Learning*, *Lecture Notes in Computer Science* **4701**, 359–370 (Springer, 2007).

Stuckenschmidt, H.; Parent, C.; Spaccapietra, S. (eds.) *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, *Lecture Notes in Computer Science* **5445**, (Springer, 2009).

Sure, Y.; Erdmann, M.; Angele, J.; Staab, S.; Studer, R.; Wenke, D. OntoEdit: Collaborative ontology development for the semantic web. In: Proceedings of the first International Semantic Web Conference ISWC 2002, *Lecture notes in computer science* **2342**, 221–235 (Springer, 2002).

Sure, Y.; Staab, S.; Studer, R. On-To-Knowledge Methodology (OTKM). In: In: Staab, S.; Studer, R. (eds.) *Handbook on Ontologies*. 117–132 (Springer, 2004).

Suyama, A.; Negishi, N.; Yamaguchi, T. CAMLET: A platform for automatic composition of inductive learning systems using ontologies. In: Lee, H.-Y.; Motoda, H. (eds.) *PRICAI 98: Topics in Artificial Intelligence 5th Pacific Rim International Conference on Artificial Intelligence Singapore, Proceedings*, *Lecture Notes in Computer Science* **1531**, 205–215 (1998).

Swartout, B.; Ramesh, P.; Knight, K.; Russ, T. Toward Distributed Use of Large-Scale Ontologies. In: *Proceedings of AAAI Symposium on Ontological Engineering*. 138–148 (1997).

Taylor, C. F.; Field, D.; Sansone, S.-A.; Aerts, J.; Apweiler, R.; Ashburner, M.; Ball, C. A.; Binz, P.-A.; Bogue, M.; Booth, T.; Brazma, A.; Brinkman, R. R.; Michael Clark, A.; Deutsch, E. W.; Fiehn, O.; Fostel, J.; Ghazal, P.; Gibson, F.; Gray, T.; Grimes, G.; Hancock, J. M.; Hardy, N. W.; Hermjakob, H.; Julian, R. K.; Kane, M.; Kettner, C.; Kinsinger, C.; Kolker, E.; Kuiper, M.; Novere, N. L.; Leebens-Mack, J.; Lewis, S. E.; Lord, P.; Mallon, A.-M.; Marthandan, N.; Masuya, H.; McNally, R.; Mehrle, A.; Morrison, N.; Orchard, S.; Quackenbush, J.; Reecy, J. M.; Robertson, D. G.; Rocca-Serra, P.; Rodriguez, H.; Rosenfelder, H.; Santoyo-Lopez, J.; Scheuermann, R. H.; Schober, D.; Smith, B.; Snape, J.; Stoeckert, C. J.; Tipton, K.; Sterk, P.; Untergasser, A.; Vandesompele, J.; Wiemann, S. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. *Nature Biotechnology* **26**, 889–896 (2008).

The BioMoby Consortium. Interoperability with Moby 1.0-it's better than sharing your toothbrush! *Briefings in Bioinformatics* **9**, 220–231 (2008).

Trajkovski, I.; Lavrač, N.; Tolar, J. SEGS: Search for enriched gene sets in microarray data. *Journal of Biomedical Informatics* **41**, 588–601 (2008).

Tsoumakas, G.; Katakis, I. Multi Label Classification: An Overview. *International Journal of Data Warehouse and Mining* **3**, 1–13 (2007).

Uschold, M.; King, M. Towards a methodology for building ontologies. In: *Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95* (1995).

Vanschoren, J.; Blockeel, H.; Pfahringer, B.; Holmes, G. Experiment databases: Creating a new platform for meta-learning research. In: Brazdil, P.; Bernstein, A.; Hunter, L. (eds.) *Proceedings of the Second Planning to Learn Workshop (PlanLearn) at the ICML/COLT/UAI 2008*. 10–15 (2008).

Vanschoren, J.; Blockeel, H.; Pfahringer, B.; Holmes, G. Experiment databases - a new way to share, organize and learn from experiments. *Machine Learning* **87**, 127–158 (2012).

Vanschoren, J.; Soldatova, L. Exposé: An ontology for data mining experiments. In: Hilario, M.; Lavrač, N.; Podpečan, V.; Kok, J. (eds.) *ECML/PKDD 2010 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-10)*. 31–44 (2010).

Vavpetič, A. The Use of Ontologies as Background Knowledge in Data Mining. Diploma Thesis (University of Ljubljana, 2011).

Vavpetič, A.; Panov, P.; Kranjc, J.; Lavrač, N. *Data mining topic ontology learning.* (Jožef Stefan Institute, Ljubljana, Slovenia, 2011).

Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; Blockeel, H. Decision trees for hierarchical multi-label classification. *Machine Learning* **73**, 185–214 (2008).

Witten, I. H.; Frank, E. *Data Mining: Practical Machine Learning Tools and Techniques.* (Morgan Kaufmann, 2005)

Xiang, Z.; Courtot, M.; Brinkman, R.; Ruttenberg, A.; He, Y. OntoFox: Web-based support for ontology reuse. *BMC Research Notes* **3**, 175 (2010).

Yang, Q.; Wu, X. 10 challenging problems in data mining research. *International Journal of Information Technologies and Decision Making* **5**, 597–604 (2006).

Žáková, M.; Kremen, P.; Železný, F.; Lavrač, N. Automating knowledge discovery workflow composition through ontology-based planning. *IEEE Transactions on Automation Science and Engineering* **8**, 253–264 (2010).

Žáková, M.; Kremen, P.; Zelezny, F.; Lavrač, N. Planning to learn with a knowledge discovery ontology. In: Brazdil, P.; Bernstein, A.; Hunter, L. (eds.) *Proc. 2ndt Intl. Planning to Learn Workshop (PlanLearn) at the ICML/COLT/UAI 2008*. 29–34 (2008).

Žáková, M.; Podpecan, V.; Železný, F.; Lavrač, N. Advancing data mining workflow construction: A framework and cases using the orange toolkit. In: Podpečan, V.; Lavrač, N.; Kok, J.; de Bruin, J. (eds.) *Proceedings of Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery*. 39–52 (2009).

Ženko, B. *Learning predictive clustering rules*. PhD thesis (University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia, 2007).

Ženko, B.; Džeroski, S. Learning classification rules for multiple target attributes. In: Washio, T.; Suzuki, E.; Ting, K. M.; Inokuchi, A. (eds.) *PAKDD*, *Lecture Notes in Computer Science* **5012**, 454–465 (Springer, 2008).

Index of Figures

Index of Tables

Appendix 1: Bibliography

Publications related to this thesis

Original scientific article (1.01)

- Panov, P.; Soldatova, L. N.; Džeroski, S. Towards an ontology of data mining investigations. *Lecture Notes in Computer Science* **5808**, 257–271 (2009).

  – This article describes the basic design principles presented in Chapter 4 and a contains brief overview of a preliminary version of the ontology.

- Stojanova, D.; Panov, P.; Gjorgjioski, V.; Kobler, A.; Džeroski, S. Estimating vegetation height and canopy cover from remotely sensed data with machine learning. *Ecological Informatics* **5**, 256–266 (2010). IF=1.351 (Joined first authorship Stojanova and Panov)

  – This article is used in two use cases presented in Chapter 9.

- Panov, P.; Soldatova, L.; Džeroski, S. Generic Ontology of Data Mining. *Journal of Web Semantics* (2012) (Revised version under revision) IF=2.789

  – This article contains the description of the OntoDM-core module presented in Chapter 6 and four use cases of the ontology presented in Chapter 9.

Published scientific conference and workshop papers (1.08)

- Panov, P.; Džeroski, S.; Soldatova, L. N. OntoDM: An ontology of data mining. In: *ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops.* 752–760 (IEEE Computer Society, 2008).

- Panov, P.; Soldatova, L.; Džeroski, S. OntoDM: towards an ontology od data mining investigations. In: Podpečan, V.; Lavrač, N.; Kok, J.; Bruin, J.(eds.) *Third generation data mining: towards service-oriented knowledge discovery, SoKD'09.* 114–118 (2009).

- Soldatova, L.; Džeroski, S.; Panov, P. Relations for information entities. In: *Proceedings of Bio-Ontologies SIG, co-located with ISMB 2012* (2012) (to appear)

Book chapter (1.16)

- Panov, P.; Soldatova, L.; Džeroski, S. Representing entities in the OntoDM data mining ontology. In: Džeroski, S.; Goethals, B.; Panov, P. (eds.) *Inductive Databases and Constraint-Based Data Mining.* 27–58 (Springer New York, 2010).

Book (editorial)

- Džeroski, S.; Goethals, B.; Panov, P (eds.) Inductive Databases and Constraint-Based Data Mining. (Springer, 2010)

Publications not related to this thesis

Original scientific article (1.01)

- Panov, P.; Džeroski, S. Combining Bagging and Random Subspaces to Create Better Ensembles. *Lecture Notes in Computer Science* **4723**, 118–129 (2007).

Published scientific conference and workshop papers (1.08)

- Panov, P.; Džeroski, S.; Blockeel, H.; Loškovska, S. Predictive data mining using itemset frequencies. In: *Zbornik 8. mednarodne multikonference Informacijska družba IS 2005.* 224–227 (2005).

- Taškova, K.; Panov, P.; Kobler, A.;, Džeroski, S.; Stojanova, D. Predicting forest stand properties from satellite images with different data mining techniques. In: *Zbornik 9. mednarodne multikonference Informacijska družba IS 2006.* 259–262 (2006).

- Stojanova, D.; Panov, P.; Kobler, A.; Džeroski, S.; Taškova, K. Learning to predict forest fires with different data mining techniques. In: *Zbornik 9. mednarodne multikonference Informacijska družba IS 2006.* 255–258 (2006).

- Džeroski, S.; Kobler, A.; Gjorgijoski, V.; Panov, P. Using Decision Trees to Predict Forest Stand Heght and Canopy Cover from LANDSAT and LIDAR data. In: *20th International Conference on Informatics for Environmental Protection - Managing Environmental Knowledge - ENVIROINFO 2006.* 125–133 (2006).

- Ivanovska, A.; Panov, P.; Colbach, N.; Debeljak, M.; Džeroski, S. Using simulation models and data mining to study co- existenceof GM/ non-GM crops at regional level. In: *20th International Conference on Informatics for Environmental Protection - Managing Environmental Knowledge - ENVIROINFO 2006.* 493–500 (2006).

- Mielikäinen, T.; Panov, P.; Džeroski, S. Itemset Support Queries using Frequent Itemsets and Their Condensed Representations. *Lecture Notes in Artificial Inteligence* **4265**, 161–172 (Springer, 2006).

Book chapter (1.16)

- Džeroski, S.; Panov, P.; Ženko, B. Ensemble methods in Machine Learning. In: Meyers, R. (ed.) *Encyclopedia of complexity and systems science* **6**, 5317–5325 (Springer, 2009)

Appendix 2: Biography

Panče Panov is a research assistant at the Department of Knowledge Technologies at the Jožef Stefan Institute, Ljubljana, Slovenia. His research work concerns the study of the knowledge discovery process, inductive databases and development of data mining techniques and their application to real-world problems from different areas. His current research focuses on developing a domain ontology of data mining and applications of the ontology in solving real world problems.