UNIVERSITÄT LEIPZIG

Fakultät für Mathematik und Informatik

Institut für Informatik

A Model for a Data Dictionary Supporting Multiple Definitions, Views and Contexts

Diplomarbeit

Leipzig, Mai 2004

vorgelegt von:

Kühn, Katrin
geb. am: 29.05.1979

Studiengang
Medizininformatik

Zusammenfassung

Auf dem Gebiet der Klinischen Studien sind präzise Begriffsdefinitionen äußerst wichtig, um eine objektive Datenerfassung und -auswertung zu gewährleisten. Zudem ermöglichen sie externen Experten die Forschungsergebnisse korrekt zu interpretieren und anzuwenden. Allerdings weisen viele Klinische Studien Defizite in diesem Punkt auf: Definitionen sind oft ungenau oder werden implizit verwendet. Außerdem sind Begriffe oft uneinheitlich definiert, obwohl standardisierte Definitionen im Hinblick auf einen weitreichenderen Austausch von Ergebnissen wünschenswert sind.

Vor diesem Hintergrund entstand die Idee des Data Dictionary, dessen Ziel zunächst darin besteht, die Definitionsalternativen von Begriffen zu sammeln und Klinischen Studien zur Verfügung zu stellen. Zusätzlich soll die Analyse der Definitionen in Bezug auf ihre Gemeinsamkeiten und Unterschiede sowie deren Harmonisierung unterstützt werden. Standardisierte Begriffsdefinitionen werden jedoch nicht erzwungen, da die Unterschiede in Definitionen inhaltlich gerechtfertigt sein können, z.B. aufgrund der Verwendung in unterschiedlichen Fachgebieten, durch studienspezifische Bedingungen oder verschiedene Expertensichten.

In der vorliegenden Arbeit wird ein Modell für das Data Dictionary entwickelt. Dazu wird zunächst eine detaillierte Anforderungsanalyse durchgeführt. Auf deren Grundlage werden existierende Terminologien und Medizinische Data Dictionaries untersucht. Dies führt zu dem Ergebnis, dass existierende Terminologiesysteme die Standardisierung von Definitionen behandeln und somit nicht die Erfassung und Repräsentation von Definitionsalternativen erlauben.

Das entwickelte Modell folgt dem aus der Terminologie bekannten konzept-basierten Ansatz und erweitert diesen um die Möglichkeit der Repräsentation alternativer Definitionen. Insbesondere wird hierbei angestrebt, die Unterschiede in den Definitionen möglichst genau zu explizieren, um zwischen inhaltlich verschiedenen Definitionsalternativen (z.B. sich widersprechenden Expertenmeinungen) und konsistenten Varianten einer inhaltlichen Definition (z.B. verschiedene Sichten, Übersetzungen in verschiedene Sprachen) unterscheiden zu können.

Mehrere Modellelemente widmen sich zudem der Explizierung von kontextuellen Informationen (z.B. der Gültigkeit innerhalb von Organisationen oder der Domäne zu der ein Konzept gehört), um die Auswahl und Wiederverwendung von Definitionen zu unterstützen. Diese Informationen erlauben verschiedene Sichten auf die Inhalte des Data Dictionary. Sichten werden dabei als kohärente Teilmengen des Data Dictionary betrachtet, die nur diejenigen Inhalte umfassen, die als relevant im ausgewählten Kontext spezifiziert sind.

Zusätzlich zu natürlichsprachlichen Definitionen besteht ein weiterer Beitrag zur Harmonisierung von Definitionen darin, strukturierte Definitionen abzubilden. Dies wird erstens durch einen Template-Mechanismus erreicht, sowie zweitens durch die Repräsentation von Relationen zwischen Konzepten. Der Template-Mechanismus ist so gestaltet, dass das Modell domänenunabhängig bleibt, um Anpassungen an andere Gebiete als das der Klinischen Studien möglich zu machen. Dieser Teil des Modells ist zudem prototypisch implementiert wobei gezeigt wird, inwieweit die Metaklassen-Architektur von Protégé, einem System zur Erstellung von Wissenserhebungswerkzeugen, für die Implementierung des Modells und insbesondere des entworfenen Template-Mechanismus genutzt werden kann.

Abstract

In the field of clinical trials, precise definitions of terms are crucial for guaranteeing objective and unbiased data collection and processing as well as ensuring that external experts are able to comprehend the research and interpret its results correctly. However, many clinical trials are unfortunately still lacking in this respect. Definitions are often imprecise or not even explicitly stated. Further, although standardized definitions of commonly used terms across clinical trials are desirable for reasons of facilitating the broader exchange and adoption of results, terms are currently defined differently in different trials.

Against this background the idea of the Data Dictionary was conceived. First of all, its goal is to represent the multiple alternative definitions of terms in use such that clinical trials can refer to them. Further, the analysis of definitions with regard to their similarities and differences and for purposes of their harmonization is to be supported. However, the latter should not be enforced for differences in definitions which may be justified due to different domain- or trial-specific conditions or different expert views.

In the present thesis a model for the Data Dictionary is developed. At first, the idea of the Data Dictionary is refined in a comprehensive requirements analysis. Based on this, existing terminologies and medical data dictionaries are reviewed and compared to the identified goals and requirements. This reveals that existing systems are all concerned with providing a standardized terminology and do thus not allow for the acquisition and representation of alternative definitions of terms.

The model developed adopts the concept-centered approach considered best practice in terminologies and adjusts it in order to accommodate alternative definitions. In particular, it strives to make explicit the kind of differences in definitions as far as possible, in order to distinguish between substantively different definitions (like contradictory expert opinions) and consistent variants of the same substantive definition (e.g. different points of view or translations into different languages).

In order to support the selection and reuse of definitions, as well as their comparison for harmonization, several model elements are devoted to the explication of contextual information (e.g. the organizational scope of a definition or the domain a concept belongs to). This information allows for several views on the content of the Data Dictionary. Views are coherent subsets of the content including only the content which is specified as relevant in the selected context.

Another contribution towards the harmonization and uniformity of definitions is made by providing structured definitions in addition to a pure natural language format. This is achieved by means of, first, a template mechanism and, secondly, relationships between concepts. The template mechanism further enables the model to remain domain-independent in order to make adaptation to other domains than that of clinical trials possible. A prototypical implementation of this part of the model demonstrates to which extent the metaclass architecture of Protégé, a meta-tool for creating knowledge-acquisition tools, can be exploited in order to implement the Data Dictionary model and, in particular, the devised template mechanism.

Acknowledgements

I am thankful to all those who have supported me during the development of this thesis.

In particular, I am grateful to my supervisor Dr. Barbara Heller for introducing me to this exciting topic. She has constantly challenged my thoughts and contributed numerous interesting ideas from which this thesis has benefited significantly.

In addition, I want to thank Kristin Lippoldt, Gerald Mücke, Roland Mücke, and Karin Quaas for stimulating discussions on preliminary versions of the Data Dictionary model.

I am particularly indebted to Dr. Evan Mellander who was a great support in proof-reading and improving my English.

I am very thankful to my parents for their love and support throughout my studies.

Last but not least, I thank Frank Loebe for supporting me in every respect, including countless discussions on details of the model as well as a very careful proof-reading.

Table of Contents

1 Introduction 1

1.1 Problem Area and Motivation 1

1.2 The Idea of a Data Dictionary 3

1.3 The Project Context: Competence Network for Malignant Lymphomas...... 4

1.4 Previous Work 4

1.5 Goals of this Thesis 6

1.6 Thesis Outline.. 6

2 Fundamentals.. 7

2.1 The Semiotic Triangle and Related Definitions 7

2.2 Definition of Data Dictionary and Similar Notions 9

2.3 Structural Characteristics of Terminologies.. 10

2.4 Common Issues in Building Terminological Systems 12

2.5 Definition of Context and View 13

3 Requirements Analysis.. 15

3.1 Description of Concepts 15

3.2 Alternative Definitions, Views and Contexts.. 21

3.2.1 Reasons for the Occurrence of Differences in Definitions............. 21

3.2.2 Requirements.. 26

3.3 Functional Requirements.. 28

4 Existing Systems 30

4.1 Medical Terminologies.. 30

4.1.1 ICD 30

4.1.2 SNOMED 32

4.1.3 LOINC.. 35

4.1.4 UMLS 38

4.1.5 GALEN.. 41

4.2 Medical Data Dictionaries.. 44

4.3 Situating the DD in the State of the Art 46

5 The Data Dictionary Model 48

5.1 Proposed Solution.. 48

5.2 Preliminaries.. 50

5.3 Interrelations between Terms, Concepts, and Alternative Definitions........ 52

5.3.1 Concepts and Alternative Definitions 52

5.3.2 Details of the Representation of Terms 53

5.4 Alternative Definitions 56

5.4.1 The DD as a Repository of Definitions 56

5.4.2 Structure of Definitions 58

5.5 Relations ... 65

5.5.1 Semantic Relations ... 66

5.5.2 Semantic Relations and their Validity in Alternative Definitions ... 69

5.5.3 Lexical Relations ... 71

5.6 Communities and Content Areas ... 73

5.7 Representation of Contexts and Views ... 77

5.7.1 Definition of Context and View in the DD Model ... 77

5.7.2 Model Elements Supporting the Context and View Mechanism ... 82

5.8 Searching for Concepts by Strings ... 87

6 Discussion of the Model ... 89

6.1 Evaluation against the Requirements ... 89

6.2 Reconsidering the Proposed Solution ... 91

6.3 Contexts and Views ... 93

6.4 What is „one“ Concept? ... 95

6.5 Domain-Independence and Extensibility of the Model ... 96

7 Prototypical Implementation ... 97

7.1 Architectural Design: DD-Server and Onto-Builder ... 97

7.2 Prototypical Implementation in Protégé ... 98

7.2.1 Rationale for Choosing Protégé ... 98

7.2.2 The Protégé Model ... 99

7.2.3 Implementation of the DD Model in Protégé ... 102

7.2.4 Discussion of the Prototypical Implementation ... 108

8 Conclusion ... 110

8.1 Summary of Results ... 110

8.2 Outlook for Further Developments ... 112

References ... 114

Index of Figures ... 120

Index of Tables ... 121

Appendix A – Specification of View Computation ... 122

1 Introduction

1.1 Problem Area and Motivation

In today's clinical research, clinical trials are an established method for the evaluation of new therapies as well as for comparing alternative therapies[1]. Of great importance for the success of a clinical trial is the maintenance of certain quality criteria. In recent years, great progress has been made with regard to the collection and definition of such quality criteria in that national and international standards and guidelines have been developed and partially implemented in legislation. An example of an international guideline is the "Guideline for Good Clinical Practice" (GCP) developed by the ICH (International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use) [ICH, 1996]. The GCP establishes guidelines for ethical issues (e.g. protection of patients' rights) as well as for assuring the scientific quality of the clinical trial. Particularly in relation to the latter issue, the GCP principles require that "[c]linical trials should be scientifically sound, and described in a clear, detailed protocol" ([ICH, 1996], p. 9).

Why is a clear and detailed trial protocol so crucial? For statistical experiments in general, which clinical trials are, it is very important to ensure that there is no systematic bias because this would distort the study result. In other words, the subjective judgements of individual doctors and special local conditions in individual trial centers should be eliminated to the extent possible. A clinical trial protocol defines the plan for conducting the study, data collection, and data analysis. The more precisely and the greater the detail in which the information in the protocol is specified, the lower the risk is for systematic bias because the interpretative freedom of the information in the protocol is minimized. Consequently, the quality and reliability of the study results is increased.

In particular, it follows from the principle of detailed protocol definition that all terms, processes, methods of examination and other data relevant for conducting the study must be defined precisely and in advance. Unfortunately, many current clinical trials are still lacking in this respect. Definitions are often imprecise, for instance in many cases it is left unspecified whether interval boundaries are to be included or not. Another example is that protocols often fail to define in sufficient detail the means and methods by which data should be collected and examinations conducted. In addition, technical terms are used of which the meanings are generally uncontroversial, but which are too vague with respect to the requirements for quality assurance in clinical trials. Thus, in certain borderline cases different interpretations and diagnoses are possible, which violates the principles elaborated above.

An example will further clarify the issue of precise definitions. `Remission`[2], `relapse`, and `progression` are frequently used terms, the principal meanings of which are generally accepted: `Remission` refers to the apparent curing of a disease, `relapse` means the

[1] According to Pocock, a clinical trial can be defined as a "planned experiment which involves patients and is designed to elucidate the most appropriate treatment of future patients with a given medical condition" ([Pocock, 1983], p. 1).

[2] Note that fixed-width font will be used throughout this thesis to indicate examples. Further, from chapter 5 on (in which the model is developed) it will also be used for names of model elements.

reoccurrence of a disease after remission, and `progression` is the advancement of the disease. A prerequisite for the occurrence of a relapse is a previously diagnosed remission, i.e. in the case of lymphoma[3] the apparent curing of the tumor. This entails the absence of measurable indications for the tumor, persisting for a certain period of time and confirmed in repeated examinations. Although these general definitions and correlations are uncontroversial among experts, they are too vague to be used in a clinical trial because they do not explicitly state the specific conditions under which a patient's individual examination data are to be diagnosed as `relapse`, `progression` or `remission`. In order to satisfy the abovementioned quality criteria regarding the elimination of bias, it must be precisely specified e.g. which examinations are to be carried out to confirm the presence of a tumor, and the length of time during which symptoms must be absent in order to constitute a remission. In particular, the necessary examinations are not specified in sufficient detail in many trial protocols.

Furthermore, precise and explicit definitions are critical in order to ensure the correct interpretation of trial results and to make possible their comparison with results of other studies. As an example, it is of little use to compare the results of two trials on the efficacy of therapies if it is unclear or even unknown how these results were obtained and which definitions underlie them. A remission rate of 40 % for therapy T1, investigated in study S1, compared to a remission rate of 35 % for therapy T2 in study S2 seems to be evidence for the higher efficacy of T1. However, it is possible that the remission criteria in study S2 are defined more strictly than in S1 and thus caused the lower rate of 35 %, while therapy T2 is actually more effective. It is therefore essential to consider underlying definitions in order to correctly assess study results.

To sum up, one can say that *explicit and precise definitions* are necessary for two main reasons:

- firstly, within a clinical trial, in order to ensure the unbiased collection of data and thus the quality of results; this holds in particular for multicenter trials in which several trial centers cooperate, leading potentially to more differences in the collection and interpretation of data;
- secondly, for cross-trial comparisons, in order to know the conditions under which the results of different trials are comparable.

While precise and explicit definitions are an important contribution towards improving the quality of clinical trials, we have not mentioned a second crucial aspect so far: many terms are defined and used differently in clinical trials and the course of trials are documented differently (e.g. documentation of measurement values in different units, different words for describing the same facts, usage of different classifications, documentation of different content, documentation at different levels of detail, different coding of data, etc.). In [Cheson, Horning, et al., 1999], for example, the differences in the definitions of remission criteria are analyzed.

The *harmonization of definitions and documentation* would be advantageous for the following reasons:

- The comparability of the results of different trials as well as cross-trial analysis with respect to new research questions would be facilitated. This is of particular interest for the research of rare diseases, because it is difficult to recruit enough patients for a study on a rare disease. Available data should therefore be exploited to the extent possible.

[3] lymph node cancer

- Definitions as well as parts of trial protocols and case report forms could be reused when developing new trials. This lowers costs as well as the time needed for the definition of the protocol, and increases the quality of definitions and contents because they are constantly being improved on the basis of experiences from previous trials. Presently, content is partly reused within study groups, although without software support, i.e. contents are copied from existing documents and changed as needed to adapt them to the current research question. This leads again to a different (if only slightly) definition and one faces once again the problem of comparability of trial results, since the similarities and differences between the definition variants are not always easy to determine.
- The development of clinical trial software is facilitated. The precise definitions could be used as a basis for software specification. Furthermore, uniform definitions allow for the development of standardized interfaces and facilitate data exchange between trials.

1.2 The Idea of a Data Dictionary

Against the background of the issues elaborated above, the idea of developing a Data Dictionary for clinical trials was conceived at the Institute for Medical Informatics, Statistics and Epidemiology (IMISE) at the University of Leipzig [Heller, 1999]. The aim of the Data Dictionary is to support the explicit and precise definition of domain-relevant terms. As determined in section 1.1, the explicit definitions that such an approach makes possible are of benefit with respect to the quality and comparability of clinical trial results. Additionally, the Data Dictionary should support the harmonization process by

1. collecting the existing alternative term definitions,
2. supporting the analysis of the definitions with regard to their similarities and differences,
3. facilitating the cooperation of experts in achieving consensus on harmonized definitions.

However, developing uniform definitions is difficult and not always possible. There can be diverse reasons for the occurrence of differences in definitions, for example, conflicting expert opinions due to the absence of conclusive research results on the basis of which a standard could be established. Further, differences in the disease entity being studied (e.g. different kinds of malignant lymphomas) may require specific definitions which are therefore justified and cannot be harmonized. Also, some definitions may be written from a more generic point of view (e.g. definitions in international guidelines) in contrast to study-specific, highly detailed definitions tailored to a concrete study. Careful analysis is needed to identify those differences which can be harmonized, and experts from different disciplines are needed to construct definitions which can be used successfully in practice. Accordingly, harmonization should not necessarily be required in all cases, but rather all alternative definitions which are justified on the basis of differences in content should be included in the Data Dictionary so that the Data Dictionary provides a repository of the different definitions in use. While designing new clinical trials, one of these alternative definitions can be selected and referred to. Thus it is clear which definition underlies the study and the demand for explicit definitions is fulfilled.

In addition it must be taken into consideration that experts from various disciplines collaborate in clinical trials, among others medical and biometrics practitioners, documentation technicians and computer scientists. Due to this interdisciplinary arrangement it is necessary to consider the various points of views of these experts in constructing the DD. Particularly interesting in this connection are various access rights with respect to the security of contents as well as the context-dependent presentation of information for the selection of relevant contents in order to avoid information overload.

The idea of the Data Dictionary is elaborated by the research group Onto-Med[4] and applied in a project which is briefly introduced in the next section.

1.3 The Project Context: Competence Network for Malignant Lymphomas

With regard to quality assurance in the healthcare sector, the German Federal Ministry for Education and Research (BMBF) has been funding research networks in the medical field since 1999. The goals are to concentrate expertise from various areas as well as to facilitate the transfer of research results to the level of patient care.

One of these research networks is the Competence Network Malignant Lymphomas[5] (in German: Kompetenznetz Maligne Lymphome, KML). Malignant lymphomas are tumors in the lymphatic system, also known as lymph node cancer. In Germany, study groups with a high level of expertise in various sub-fields of lymphoma research have existed for years and have made substantial progress in the diagnosis and therapy of malignant lymphomas. Alongside the present study groups, doctors as well as various institutions and self-help organizations in the field of lymphoma research in Germany have joined forces in KML, to group together available expert knowledge, to improve the exchange of knowledge between scientists and medical doctors as well as to optimize patient care.

One sub-project of KML is sub-project 2 "Telematics and Computer-based Quality Management", which is directed by IMISE (cf. [Heller, Löffler, 2002]). The goal of this project is to develop innovative computer-science concepts and software to support quality assurance, one of which is the Data Dictionary. KML offers an optimal domain of application for the Data Dictionary. Due to the cooperation of various institutions, there is a high degree of heterogeneity in KML, so that the problems depicted in the motivation section with respect to non-uniform term definitions and documentation of data are especially relevant here. For example, each of the study groups in the KML maintains its own study database based on different data models. Different laboratories use different measuring procedures and accordingly yield results in varying units and of varying quality. The idea of the Data Dictionary depicted in section 1.2 thus supports the quality improvement and quality assurance aspired to in KML.

1.4 Previous Work

A first version of a software tool which supports the development of data dictionaries has been developed by the Onto-Med research group, namely the Onto-Builder [Heller, Lippoldt, et al., 2003a; b]. The Onto-Builder is a web-based application supporting the collaborative and distributed input of terms as well as their alternative definitions. The content, which is stored in a relational database, constitutes a Data Dictionary.

[4] http://www.onto-med.de

[5] http://www.lymphome.de

Thus far, the tool has been employed for developing data dictionaries in the domain of clinical trials on malignant lymphoma (by the Competence Network Malignant Lymphomas) as well as in the domain of cardiological trials (by the Coordination Center for Clinical Trials Leipzig[6]).

The main entities of the data model of the DD in this first version are shown in the UML[7] diagram in figure 1.1. (Attributes are omitted for the sake brevity; the classes are explained below.)

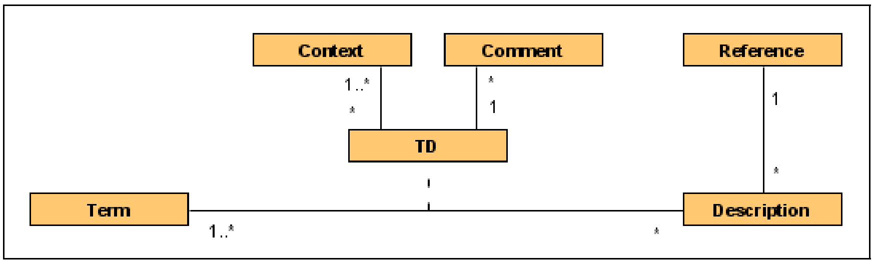


Figure 1.1: Overview of the model of the Data Dictionary Version 1.

The main focus in this first version is on the acquisition of domain-relevant terms (instances of `Term`) with their alternative definition(s) (instances of `Description`). A term may consist of one or more words, and a description is recorded as text. The association between both classes (represented by the association class `TD`) is a many-to-many association in order to accommodate synonymous terms which have the same description[8]. The class `Reference` contains the bibliographical source from which the description is cited (e.g. standard literature, study documents), or the editor's name if it is a newly created description.

The association between a term and its description is qualified by `Context`, because a term may have different meanings[9] depending on the contexts in which it is used. For example, the term `application` has a different meaning in the context of medicine than in computer science. Therefore, the class `Context` defines a coarse classification of subject areas which can be used for indicating the context in which a description of a term is valid.

The analysis of the recorded knowledge as well as the discussion of different alternative definitions is supported by providing the functionality of inserting `Comments` on a term-description pair. Further, to support the harmonization process the definition of so-called consensus suggestions is possible (represented as specially marked descriptions).

A simple versioning mechanism allows the history of the editing process for each description to be followed. Old description versions can be made visible in order to understand the changes made to the descriptions. Additionally, to keep track of which parts of the DD have been edited by which users, information about the users' editing activities is stored. The user and role concept allows for the management of several user groups with different rights to access certain functionalities: readers, editors, moderators and administrators.

[6] http://www.kksl.uni-leipzig.de
[7] UML = Unified Modeling Language; see section 5.2
[8] cf. section 2.1 for the definition of synonymy
[9] cf. the definition of ambiguity in 2.1

The data model described above is rather simple and exhibits several inadequacies. Firstly, it is not capable of supporting the analysis of definitions with regard to their similarities and differences. In particular, it mixes alternative definitions which refer to different meanings of an ambiguous term with definitions that describe the same meaning. Further, it is not possible to specify whether two definitions differ in content or merely in point of view. Secondly, the representation of synonymous terms poses problems in cases where there exist multiple definitions for one term. If the synonym is valid for all these alternative definitions, it must be assigned to each definition separately. This is hard to maintain, especially if new definitions are added. Thirdly, the context representation is very simple; context-based information presentation and different points of view were not implemented. Lastly, support for the harmonization of definitions is restricted by allowing for one consensus suggestion only. Instead, as emphasized in the previous section, complete harmonization may not be possible or even desirable, and thus multiple consensus suggestions for different scopes of validity may be required.

1.5 Goals of this Thesis

In this thesis the specification of the data model for the Data Dictionary Version 2 (hereafter: DD) is to be developed. This model should overcome the inadequacies of the first version described above as well as enhance the Data Dictionary’s representational and functional capabilities.

For this reason the first goal of the thesis is to conduct an analysis of requirements based on experience from the first version of the DD (cf. section 1.4) as well as a review of the literature. Here the focus is set to requirements with respect to the representation of terms with their alternative definitions, views and contexts as well as support of the harmonization process.

The core of the thesis consists in developing the data model for the Data Dictionary corresponding to the identified requirements. The goal here is to design the data model in a domain-independent way to permit adaptation to fields other than clinical trials. In addition it should be flexible and extendible, in order to allow for the accommodation of modified requirements on the contents to be saved. The realization of the data model should be sketched out with the help of a partial prototypical implementation.

1.6 Thesis Outline

After the introductive chapter 1, chapter 2 presents an overview of the field of terminology and introduces basic concepts and methodologies. Chapter 3 investigates the requirements for the model of the data dictionary, taking into account special requirements by the domain of clinical trials but not restricting it to this specific domain so that the resulting model can be extended to other domains. Chapter 4 presents and analyzes existing terminologies and medical data dictionaries with regard to the requirements. In chapter 5 the model is elaborated. An evaluation of the model against the requirements and a discussion of important model choices is given in chapter 6. A partial prototypical implementation of the model is described in chapter 7. The thesis concludes with chapter 8 in which the results are summarized and further developments and possible extensions to the model are outlined.

2 Fundamentals

This chapter sets the basis for this thesis by introducing fundamental principles and defining important notions necessary to understand the chapters which follow. The first two sections define basic notions, which are particularly necessary because the field of terminology is characterized by the occurrence of a plethora of notions defined varyingly in the literature, providing a high potential for confusion. Section 2.3 introduces the fundamental structural characteristics of terminologies, while section 2.4 highlights some issues which are commonly encountered when developing terminologies in general and medical terminologies in particular. The last section, section 2.5, defines two central notions of the title of this thesis, namely context and view.

2.1 The Semiotic Triangle and Related Definitions

Most approaches to describing the meaning of words are based on the so-called semiotic triangle – a fundamental principle in the field of terminology describing the relations between words, meanings and reality. In what follows, the semiotic triangle is defined following [DIN 2342-1, 1992] and [de Keizer, Abu-Hanna, et al., 2000].[10]

The semiotic triangle relates objects, concepts, and designations.

> **Objects** are the things that exist in reality; they can be concrete (e.g. an individual `heart`) or abstract (e.g. the `pain` of some patient). Objects can be described by their characteristics. Often, the notion of 'individual' is used synonymously to 'object'.
>
> A **concept** is a unit of thought which is formed by abstracting the common characteristics of similar objects. It is language-independent but may be influenced by social or cultural background. Example concepts are `<heart>` and `<pain>`[11]. Sometimes, in order to make clear that the concept but not an individual is meant, concepts are referred to as the 'prototypical heart' or the 'ideal heart'.
>
> **Designations** are labels which refer to concepts. A **term** is a linguistic label and may consist of one or more words. For example, the English term `heart` and the German term `Herz` designate the concept `<heart>`. A **code** is a special label used within computer systems to identify concepts and consists of numbers and / or alphabetic characters.

Note that according to the above definition, terms are only such linguistic labels which refer to concepts. Terms should therefore not be confused with proper names, which refer to individuals, e.g. `patient John`.

The most important aspect of the semiotic triangle is the term-concept distinction, i.e. concepts as language-independent units of meaning which can be named by designations. Accordingly, terminologies which adopt this distinction (and thus define concepts instead of terms directly) are said to follow a concept-centered approach.

[10] The semiotic triangle is generally attributed in the literature to [Ogden, Richards, 1923], although there exist according to [Bürkle, 2000] other similar approaches.

[11] Although concepts are defined as being language-independent, language is needed in order to discuss them. Accordingly, concepts are therefore enclosed in angle brackets (<...>) in order to distinguish them from terms.

While the term-concept distinction is widely accepted, the exact definition of the notion of concept is very controversial. A deeper analysis of the notion of concept is beyond the scope of this thesis – many works in terminology, linguistics, philosophy and conceptual modeling have struggled with this question but no final answer has been found yet. We have chosen to give the above definition because it is a standard definition which reflects the maximal common understanding of most approaches. The disadvantage of the definition is that it is rather vague, in that it does not explain what a 'unit' of thought should be.

It is important to understand that the relation between designations and concepts is not a one-to-one relation – instead, a term can have multiple meanings, and a concept can be designated by multiple terms, as described in the following definitions.

Homonymy, Polysemy, Ambiguity. Homonymy and polysemy are two notions commonly used to describe a term having more than one meaning, i.e. a term that designates more than one concept. The difference between the two notions is that the different meanings of a polysemous term are related while in the case of a homonymous term they are unrelated. The English term `bank` is the standard example of a homonym because it can mean (among other things) a financial institution or a bench – two unrelated meanings. On the other hand, `face` is a polysemous term because its meanings as verb and noun are related. As argued by [Lyons, 1995] (p. 415), however, the difference between homonymy and polysemy cannot be defined precisely and is arbitrary. In this thesis, we will therefore use the more general notion of ambiguity to include both homonymy and polysemy.

Synonymy. Two terms are synonymous if they refer to the same concept. Synonyms can be of the same (e.g. `clinical trial`, `clinical study`) or of different languages (e.g. `informed consent`, `Einverständniserklärung`). Ideally, synonyms can be exchanged in every context without changing the meaning of a sentence. This ideal notion of synonymy is also referred to as "absolute synonymy". However, since a term may have more than one meaning as described above, absolute synonymy is very rare. The more common case of synonymy is called "quasi-synonymy" [Lyons, 1995], meaning that two terms are synonymous (and thus interchangeable) only in certain contexts, namely in those contexts where both terms have the same meaning.[12]

The occurrence of synonyms and homonyms is – alongside the occurrence of imprecise definitions as described in the introduction – the main reason for ambiguous communication. The aim of an ideal terminology is therefore to reach a one-to-one mapping of terms and concepts by defining preferred terms.

Preferred term. In order to support unambiguous communication, a one-to-one mapping between terms and concepts is desirable and is achieved by selecting one term out of the set of synonyms as the preferred term. The preferred term must be unambiguous and should always be used where unambiguous reference to a concept is necessary. Synonyms of the preferred term can still be defined in order to provide more flexibility in searches.

[12] Context is defined in section 2.5.

2.2 Definition of Data Dictionary and Similar Notions

The notion of *data dictionary* occurs with two different but related meanings in the literature: first, in the context of database systems and second, in the context of medical terminologies. In order to unambiguously refer to the second sense, the more precise term *medical data dictionary* is often used. In the following, we define both meanings.

Data Dictionary (Computer Science)

> According to [Rumbaugh, Blaha, et al., 1993], a data dictionary should be developed during object-oriented analysis of software systems in order to describe the meaning behind class names. This is necessary because isolated class names can be interpreted differently by different system developers. Rumbaugh et al. therefore recommend describing each class name in a separate paragraph, including any assumptions about the problem domain as well as associations, attributes and operations. Definitions take the form of narrative text, i.e. they contain little structure.
>
> In a narrower sense, a data dictionary is part of a database system and contains the database schemata for all database(s) within the database system (cf. [Balzert, Helmut, 2000], p. 721).

Medical Data Dictionary

[Bürkle, 2000] provides a comprehensive and current overview of medical data dictionaries. According to him, there is no widely accepted, uniform definition of a medical data dictionary. He analyzes different existing definitions and prefers the definition initially developed by Prokosch, Dudeck and Michel:

> “A medical data dictionary is a central thesaurus for the controlled definition of the medical vocabulary to be applied in a hospital information system (HIS), which is also capable to represent the semantic relationships existing between all HIS objects and to link the local vocabulary to standardized international nomenclatures and knowledge sources.”
> ([Bürkle, 2000], p. 17)

The definition of medical data dictionary above includes the notions of vocabulary and nomenclature. In the literature, many further notions like terminology, glossary, lexicon, and controlled vocabulary also occur frequently and are used interchangeably in order to refer to systems which define the meaning of terms. As noted by de Keizer et al., “literature on terminological systems […] is hard to understand and is ambiguous […] due to heterogeneity and indistinctness in the terminology used to describe the terminological systems” ([de Keizer, Abu-Hanna, et al., 2000], p. 16). After an extensive review of the literature, de Keizer et al. propose definitions for these notions, which are adopted here ([de Keizer, Abu-Hanna, et al., 2000], p. 18). They consider terminology as the most general term which is defined as follows:

> “A **terminology** is a list of terms referring to concepts in a particular domain.”

Each terminology can have one or more of the following additional characteristics:

> “A **thesaurus** is a terminology, in which terms are ordered, e.g. alphabetically or systematically and in which concepts can possibly be described by more than one (synonymous) term. When a concept in a terminology or thesaurus is accompanied by a

> definition, it is called a **vocabulary** or **glossary**. A **nomenclature** is a system of terms composed according to pre-established composition rules or the set of rules itself for composing new complex concepts. **Classification** is an arrangement of objects or concepts (by the is_member_of relation) based on their essential characteristics into groups of concepts, called classes. [...] A terminology, thesaurus, vocabulary, nomenclature or classification is called a **coding system** when the system uses codes for designating concepts."
> ([de Keizer, Abu-Hanna, et al., 2000], p. 18)

The difference between a classification and a nomenclature is further clarified by [Leiner, Gaus, et al., 1999], who emphasize that, when using a classification, each object is assigned to exactly one class. It is thus important for a classification that the classes are disjoint (i.e. not overlapping), in order to ensure that each object can be classified unambiguously. In contrast, by using a nomenclature objects can be described by assigning all appropriate concepts, which are not necessarily disjoint.

After the requirements analysis in chapter 3 and the overview of existing terminological systems in chapter 4, the relation of the Data Dictionary to the above notions is described in section 4.3.

2.3 Structural Characteristics of Terminologies

In principle, concepts can be defined either intensionally or extensionally (cf. [de Keizer, Abu-Hanna, et al., 2000]):

> An **intensional definition** of a concept describes the properties that an object must have in order to belong to that concept. This set of properties is the intension of the concept.
>
> An **extensional definition** of a concept enumerates all the objects which belong to the concept. This set of objects is called the extension of the concept.

The concepts in a terminology[13] are often organized hierarchically, where two types of hierarchies can be distinguished (cf. [Leiner, Gaus, et al., 1999], p. 38):

> In a **monohierarchy**, each concept has at most one direct superordinate concept.
>
> In a **polyhierarchy**, multiple direct superordinates per concept are allowed. A polyhierarchy must not contain cycles, i.e. it is an acyclic directed graph.

Both types of hierarchy are illustrated in figure 2.1 below.

[13] As shown in chapter 4, not all terminologies adopt the term-concept distinction and contain terms only without making the underlying concepts explicit. Thus, such terminologies actually organize the terms hierarchically instead of the concepts. However, in order to avoid confusion we will always refer to concepts when describing the general characteristics of concept representation in terminologies, keeping in mind that many systems do not make them explicit and thus may represent these characteristics for the designating terms instead.

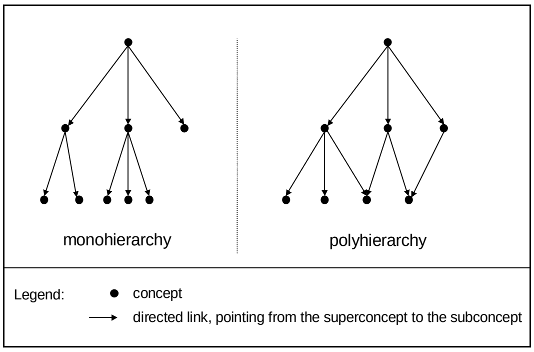

monohierarchy
polyhierarchy
Legend:
● concept
⟶ directed link, pointing from the superconcept to the subconcept

Figure 2.1: Monohierarchy vs. polyhierarchy.

The meaning of the hierarchical relation differs from terminology to terminology, and often even within one hierarchy. Two hierarchical relations which are often used are the is-a and the part-of relation. Like many other notions introduced thus far, these relations are also defined varyingly in the literature – for example, an extensive discussion of the meaning of the is-a relation is contained in [Brachman, 1983]. For the purpose of this thesis, it is unnecessary to provide an extensive discussion of the specific details and differences discussed in the literature. The following explanations rather than definitions are given according to [de Keizer, Abu-Hanna, et al., 2000] and should suffice to convey an understanding of the principal meaning of the two relations.

> A subconcept **is-a** superconcept if all objects which are contained in the extension of the subconcept are contained in the extension of the superconcept as well. From an intensional point of view, the superconcept is an abstraction of its subconcepts in that it contains properties which are common to all of its subconcepts. The is-a relation is a relation between concepts and should be distinguished from the relation between objects and concepts.
> Example: `<multicenter clinical trial> is-a <clinical trial>`
>
> The **part-of** relationship is also a hierarchical relation where the superordinate concept is a whole and the subordinate concepts represent its parts.
> Example: `<arm> part-of <body>`

The analysis of existing systems in chapter 4 shows that many terminologies do not define the meaning of the relation used for establishing a hierarchy and mix several organizational principles (e.g. is-a, part-of, and any other purpose-dependent criteria for grouping “similar” concepts).

Concept hierarchies are related to **inheritance** mechanisms. Inheritance mechanisms exploit the is-a relation and allow for the assignment of a property at the top-most possible concept in the hierarchy, which is then also a (inherited) property of all subconcepts. Thus, inheritance avoids the redundant specification of information. According to the types of hierarchies introduced above, i.e. monohierarchy and polyhierarchy, inheritance can be distinguished into single inheritance and multiple inheritance approaches. The latter needs to specify mechanisms which resolve conflicts that may occur if the information that a concept inherits from its multiple superconcepts is inconsistent. Details on inheritance mechanisms can be found in [Luger, 2001].

A further structural characteristic of terminologies are the semantic dimensions or axes along which concepts are described and organized.

> A **multiaxial** terminology consists of two or more independent axes, where each axis contains concepts which are described and organized hierarchically according to a different semantic perspective. Concepts from the different axes can be combined in order to form complex concepts.

An example of a multiaxial terminology is SNOMED [SNOMED International, 2004] which is a multiaxial nomenclature describing diseases as complex concepts by combining concepts from eleven axes, e.g. topology and morphology (cf. section 4.1.2). Note that monoaxial terminologies, i.e. systems with only one axis, do not necessarily use only one semantic perspective. Instead, different semantic perspectives are often mixed within the single hierarchy of such systems.

2.4 Common Issues in Building Terminological Systems

Combinatorial Explosion. For large domains like that of medicine, it is impossible to enumerate all relevant concepts because every different combination of characteristics constitutes a different concept. As an example consider the concept `<clinical trial>` which has as subconcepts, for example, `<monocenter study>`, `<multicenter study>`, `<cardiological study>`, `<oncological study>`. Combinations of these concepts are also possible leading to concepts such as `<multicenter cardiological study>`, `<monocenter cardiological study>`, and so on. In order to avoid combinatorial explosion, methods for compositional concept representation – i.e. defining *complex concepts* on the basis of elementary concepts – are important. One approach to complex concepts is mentioned in the definition of multiaxial terminology (section 2.3) where the elementary concepts are enumerated and organized into axes, and complex concepts can be defined by combining concepts from the different axes. An overview of further approaches is given in [Spackman, K. A., Campbell, 1998].

Tangled Hierarchies. As mentioned in the previous section, one can often notice that different relationships are mixed within is-a hierarchies. This holds in particular for mixtures of is-a with part-of. Further, non-hierarchical relations are partially combined with hierarchical ones, or different semantic perspectives are introduced without a clear, explicit distinction. Tangled hierarchies can lead to incorrect inheritance and pose a problem with regard to maintaining consistency and reusability. (cf. [Rector, A. L., Rogers, 1999]).

Delimitation of Objects and Concepts. Although the distinction between concepts and objects as defined in the semiotic triangle (introduced in section 2.1) may seem clear cut, this is not the case in practice. This is to some extent a controversial philosophical question (Is a specific concept itself an object in the real world?) but also a practical question in the construction of terminologies. Many representation languages model concepts as classes and objects as instances. However, the leaves of the concept hierarchy are often represented as instances which becomes problematic if the terminology is to be reused in an application where a higher level of detail is required. Further, "undesirable ambiguities and misinterpretations of represented knowledge" arise ([Welty, Ferrucci, 1994], p. 1). For further discussion on this problem the reader is referred to [Welty, Ferrucci, 1994] and [Baader, Calvanese, et al., 2003] (p. 363).

Complexity of the Medical Domain. The complexity of the medical domain makes the development and maintenance of medical terminologies more difficult than for most other domains. In particular, the size of the domain, the changing nature of medical knowledge as research progresses, and the many specialized areas contribute to the high complexity. Regarding the size of medical terminologies, Rector states that "[t]he smallest useful medical terminologies contain on the order of 10,000 concepts" and several systems contain on the order of 200,000 concepts ([Baader, Calvanese, et al., 2003], p. 425).

Modeling of Abnormalities. A specific problem which adds to the abovementioned complexity of the medical domain is the problem of modeling abnormalities and exceptions. For example, although most humans have two kidneys, it is also possible to live with only one. Further examples are given by Rector who emphasizes that although each individual abnormality is rare, "many combinations of abnormalities are possible" and "[t]aken collectively, they are surprisingly common" meaning that many patients are atypical in some respect ([Baader, Calvanese, et al., 2003], p. 432). Apart from the huge number of possible abnormalities that need to be represented, the modeling of abnormalities has serious implications for methods of inheritance because exceptions occur so frequently.

There are many more issues which could be discussed at this point, but those described above are the most important and most frequently occurring problems in the development of medical terminologies. Further detailed discussions can be found for example in [Rector, A. L., 1999] and [Baader, Calvanese, et al., 2003] (in particular chapter 13 on medical terminologies written by A. Rector). In this thesis, further issues related to alternative definitions, views and contexts are discussed in detail in the requirements analysis (section 3.2).

2.5 Definition of Context and View

Context and view are notions which occur in the title of this thesis. Since these notions are used very differently in a broad variety of research fields (linguistics, philosophy, terminology, computer science), this section introduces the definitions which are the basis of our understanding of contexts and views.

Generally, two meanings of context can be distinguished. The first meaning refers to the **linguistic context**, which is the surrounding text of a word or utterance. Secondly, the notion of context is also used to refer to the **situative context**, which includes any characteristics of the situation in which an utterance occurs, e.g., time, space, surrounding events, knowledge of the persons, cultural environment, conventions, etc. (cf. [Lyons, 1995], p. 422). Although numerous other typologies and definitions of context have been proposed (cf. [Bouquet, Serafini, et al., 1999]), these two types are generally accepted.

Within this thesis, if not stated otherwise, we use the word **context** as the general notion including both of the abovementioned types of context. This is considered useful because both types of context are of importance with regard to the relation between words or terms and concepts, i.e. both types of context contribute to the disambiguation of words. Situative context is especially interesting for information retrieval systems. Such systems usually have to deal with huge amounts of information, and queries to the system (usually just a set of keywords) often return much information, most of which is irrelevant for the user. Hence, a focus of research is on the development of methodologies for context-based information presentation, i.e. exploiting information about the situative context of the user (e.g. the current task) in order to provide information which is most relevant to the user’s immediate needs. However, linguistic context can also be used in information retrieval for interpreting the query. For example, information about frequent cooccurrence of concepts can be used to disambiguate keywords in the query.

The understanding of **view** in this thesis is grounded on the two following similar definitions. In relational database systems, views are used for selecting a subset of the data in order to provide different user groups with possibly different access rights for certain content. In the specification of the Unified Modeling Language, which is used for modeling software systems, a view is defined as “[a] projection of a model, which is seen from a given perspective or vantage point and omits entities that are not relevant to this perspective” ([OMG, 2003a] p. Glossary-16). Both definitions have in common that views are a means to select certain aspects of some underlying base model.

3 Requirements Analysis

In this chapter, the requirements of a data dictionary for clinical trials will be elaborated. The aim is to analyze the specific requirements of this domain without restricting the model to it, i.e. we are aiming at a model which can easily be extended to other domains.

In the introduction, we have already identified the main goals for the content that should be available in the DD. It must be capable of:

1. storing concept definitions,
2. representing alternative definitions, contexts and views,
3. supporting harmonization without making it mandatory.

Each of these main objectives is rather general and implies numerous detailed requirements which will be introduced and explained in the following sections. The requirements are derived from the experience gained by the existing first version of the DD (cf. section 1.4), by discussion with domain experts and by analyzing example definitions, as well as from the literature. With regard to the literature, it should be noted that a unified and detailed set of requirements has not been established until now. Although several works exist, they are mostly very specific to a problem domain and anticipated use of the system, or, if they are generic, the requirements are vague and less detailed, and thus insufficient for our requirements analysis. Further, while the first of the above three objectives is mostly covered by existing works, the latter two are hardly addressed.

The requirements analysis is structured as follows: we first address the question of which elements are needed to describe concept definitions in the DD (section 3.1). Secondly, in section 3.2, the requirements with regard to alternative definitions, views, contexts and harmonization of content are presented. In the last section (3.3), the functional requirements with regard to editing DD content are covered.

3.1 Description of Concepts

This section concentrates on specifying all those elements which are needed for representing *one* concept definition in the DD (e.g. source of the definition, synonyms, etc.). It abstracts away from the needs that arise from alternative definitions and views or contexts, i.e. in this section we try to isolate the “pure” content requirements, assuming there were neither alternative definitions nor view- or context-dependent information. The requirements given in this section could therefore be compared to those of “traditional” medical terminologies, which generally have not taken into account requirements concerning alternative definitions, contexts and views.

For clarity, the requirements are enumerated (e.g. “R1” for “Requirement 1”). If appropriate, examples are given to illustrate important aspects of a requirement (“R1-Ex2” denotes the second example for R1).

R1 Definitions

The DD should contain precise natural-language definitions of domain-relevant terms. If definitions are cited from the literature or any other official documents, their source should be referenced.

With respect to the format of definitions, simple layout and formatting options should ideally be available, in particular: simple tables, bold / italic / underlined font, as well as the inclusion of graphics.

R1-Ex1 Examples of domain-relevant terms with regard to the DD for the KML: `relapse, remission, response criteria, clinical trial, malignant lymphoma, leukocyte, standard operating procedure, informed consent, B-symptoms`

R1-Ex2 Example of a definition ([ICH, 1996], p. 6): "`Multicentre Trial: A clinical trial conducted according to a single protocol but at more than one site, and therefore, carried out by more than one investigator.`"

Although the requirement of natural-language definitions might seem like a trivial requirement for a terminology, it is not at all common in existing systems. Many systems do not provide explicit natural language definitions but rather define the concept by means of its place in the hierarchy and / or by relations to other concepts. However, this often leaves implicit many aspects of the concept definition which are essential with regard to shared understanding and semantic transparency. Also, the anticipated use of the DD as a glossary requires natural language definitions.

R2 Templates

The DD should provide the possibility to define templates for the definition of similar concepts (e.g. `laboratory data`, `processes`, `diseases`, `classifications`). In the following, we will call such a group of similar concepts a **category**. The purpose of templates is to achieve uniformity of definitions across similar concepts and across different editors, which will facilitate the comparison of definitions.

R2-Ex1 An example of a template for a structured definition is given below, where three concepts of the category `end point` are defined according to the following structure: `Response category: … Definition: … Point of Measurement: …` (The definitions below are cited from [Cheson, Horning, et al., 1999], p. 1249).

End Point	Response Category	Definition	Point of Measurement
Overall survival	All patients	Death from any cause	Entry onto trial
Event-free survival	Complete remission, Uncertain complete remission, Partial remission	Failure or death from any cause	Entry onto trial
Disease-free survival	Complete remission, Uncertain complete remission	Time to relapse	First documentation of response

From the above example, we can derive the following requirements for such a template. It may consist of named sections. In the above example, the template consists of three sections, namely Response category, Definition, and Point of Measurement. Further, it might be useful to support the entry of definitions by restricting the allowed values for a template section. Conceivable constraints are restrictions to certain value types (integer, string, graphics, etc.) or to concepts of certain categories. For an example of a restriction to categories consider the column response category which could be restricted to allow only concepts belonging to the category remission criteria (e.g. for event-free survival, the template section for response category is filled with three concepts all of which are remission criteria: complete remission, uncertain complete remission, partial remission).

The selection of a category for a concept should be optional since it cannot be assumed for every concept that an appropriate category with a template exists.

R3 Synonyms

The DD should provide the possibility of capturing synonyms.

R3-Ex1 [Ger.] Leukozyten, Leukos

[En.] leukocytes, white blood count

For each language, one term out of the set of synonyms may be labeled as the preferred term for this language.

R4 Abbreviations

The DD should manage abbreviations of terms. Note that it is conceivable that the same string is both an abbreviation of another term as well as a term in its own right, although such cases should be very rare.

R5 Multilinguality

The DD should be multilingual, i.e. translations of content should be available where possible (e.g. English or German translations of terms and definitions). Multilingual content is necessary because multicenter studies are increasingly being conducted internationally, leading to more and more international cooperation between experts and institutions from different nations.

It should be indicated clearly if two definitions (or any other content) are translations of the same content. Consider for example definitions in the Guideline for Good Clinical Practice [ICH, 1996], which is originally written in English, but for which a German translation exists. It must be clear that such definitions are merely translations into a different language – so that they are not confused with substantively different definitions.

Note that translations should not be mandatory. Content should be entered as it is available from the relevant sources. The higher priority is to achieve high-quality definitions, while the lower priority is to achieve complete translations into other languages. Thus, until complete translation of the whole DD is available, viewing the DD in different languages will be a restricted feature because it will not show the complete content.

Note that it is a common practice to not translate English specialist terms, but rather to use them in German as well (e.g. `bulk`), often in parallel to German equivalents. The same holds for abbreviations (see example R5-Ex1 below). The model should be capable of representing this.

R5-Ex1 For the German term `Standardarbeitsanweisung`, the English abbreviation `SOP`, which stands for `standard operating procedure` is commonly used.

R6 Relations

In the DD, it should be possible to define relations. Here it is useful if types of relations (R6-Ex1) can be defined and later selected when defining a relation between concrete concepts (R6-Ex2).

R6-Ex1 `synonym-of`, `part-of`, `is-a`, `causes`, `measures`

R6-Ex2 defining a relation between concepts using the relation type `symptom-of`:

```
<concept₁>   symptom-of   <concept₂>
<concept₁>   symptom-of   <concept₃>
```

A very desirable feature would be to restrict the concepts that may be linked by a relation type to certain categories of concepts. Regarding the allowed arity of relations, it should be noted that relations of arbitrary arity are conceivable but binary and ternary relations (i.e. relations with two or three arguments) are of greatest practical relevance. Many systems support binary relations only, since the complexity of representing ternary relations is considerably higher (cf. [Rumbaugh, Jacobson, et al., 1999]).

R7 Rules

The DD should provide a means for representing rules. According to [Bürkle, 2000] (p. 7), the implementation of knowledge-based functions using rules for the assessment of patient data is an important criterion for the quality of medical information systems. Such rules can either be used for decision support (e.g. automated evaluation of laboratory data) or for monitoring (i.e. to generate warnings). Furthermore, in the documentation of clinical trials rules are used for plausibility control of data, i.e. testing for improper combination of data values (cf. [Leiner, Gaus, et al., 1999], p. 134).

Clinical trial protocols and standard operating procedures contain many rules describing instructions, consequences, etc. However, they are mostly contained in narrative text, and thus they are difficult to extract. Explicit representation of rules in a structured way helps to capture the essence of the rule by eliminating textual elements that only hinder conciseness and is a good preparation towards the formal representation of rules.

The following examples of rules are cited from the study protocol DSHNHL 1999-1A [Pfreundschuh, Trümper, et al., 2001]. Note that none of these rules are marked as a rule in the protocol.

R7-Ex1 “radiation dose and dosage schedule

If the area to be treated is large, e.g. in the abdomen, the individual dose may be reduced to a minimum of 1.4 Gy.” (p. 47)

R8 Additional Knowledge

The DD should – apart from the definition of a term – represent additional knowledge which is related to a term.

Such knowledge may be entered as free text or in the form of associated relevant documents (e.g. relevant standard operating procedures, or clinical guidelines). Additional knowledge may be linked to multiple concepts, which then act as indices to that knowledge.

R8-Ex1 Example of a definition and additional knowledge:

Concepts which this additional knowledge could be linked to are underlined. (Definitions are taken from [Diehl, Sieber, et al., 2001], translated by the author.)

Definition:

“A relapse is defined as the occurrence of new or the reappearance of initial tumor lesions or B-symptoms after complete remission (CR) 3 months after completion of therapy at the earliest. If the interval is shorter the case is assessed as progression.” (p. 73)

Additional knowledge*:*

“In principle, in the occurrence of a relapse a complete staging with new histological confirmation is to be conducted. All cases of relapse are assessed by the reference radiology and reference pathology. A precondition for radiotherapeutical assessment is the presence of all the following documents:

1. Immediate documentation and notification of the relapse and the relapse localization by means of the case report form ‘Follow-up’;
2. Radiotherapy plan and verification images of the initial radiotherapy are to be sent, if not yet done so, to the radiotherapy reference center.
3. The documentation of the relapse through imaging methods (X-raying, CT, NMR) is sent to the radiotherapy reference center.“ (p.76)

Furthermore, therapy recommendations for relapse are given in the protocol, in which `progression` or `early relapse` (relapse within one year after completion of therapy) is differentiated from `late relapse` (more than one year after completion of therapy). From this example, it seems desirable to have a link like: `For further / related information see early relapse and late relapse`.

R9 Classifications

Aside from definitions of domain-relevant terms (cf. R1), the DD should also provide existing classifications, e.g. `WHO classification of Malignant Lymphoma`, `Common Toxicity Criteria (CTC)`, `performance status according to the Karnofsky index`, `performance status according to EGOG scale`.

See also the functional requirement R20 regarding the connection to external, electronically available terminological resources. They may contain such classifications.

R10 Versioning

The DD should support the versioning of definitions.

Due to progress in medicine it is necessary to change or expand the content of existing definitions. In doing so, however, it is necessary according to [Cimino, 1998] to preserve concept permanence, or the requirement that “the meaning of a concept, once created, is inviolate” ([Cimino, 1998], p. 396). This is necessary to ensure that documents or applications containing references to definitions in the DD continue to maintain their validity. Hence definitions cannot simply be modified, but rather must be versioned, enabling applications to make unambiguous reference to specific versions of definitions. Contents should therefore not be deleted, but rather marked as invalid or obsolete in order to prevent their use in newly created documents while at the same time preserving their availability for older documents which contain references to them.

R11 Semiformal Content

The content in the DD should be represented on a semiformal level.

A semiformal level is understood as a mixture of natural language contents which cannot be interpreted by a computer, and formal contents that can be processed automatically. The first are necessary, e.g. for storing definitions (cf. R1), while formally represented aspects (e.g. relations) are also desirable, since they can be interpreted by machine and furthermore are not subject to the ambiguity of natural language.

Note that redundancy inevitably occurs if the same information is represented on more than one formalization level and must therefore be accepted as long as no automatic translation between both formalization levels exists. Experts should, however, be in a position to verify consistency. An example of a possible redundancy are relations which are explicitly represented in the DD (cf. R6), but may also be described informally as part of textual definitions. Further examples for semiformal components are templates (R2) and the representation of rules (R7).

3.2 Alternative Definitions, Views and Contexts

In this section, we describe the requirements regarding alternative definitions, views and contexts. The requirements specified in the previous section have not thus far considered these issues on the assumption that the collaborating experts involved in the construction of the DD were able to agree on contents and definitions. As was determined in the introduction, however, consensus on definitions is often impossible, unrealized or undesired. This becomes obvious when one looks up the definition of a term in various sources and compares them; the same term is often defined and used differently.

Before analyzing the requirements for the representation of alternative definitions, views, and contexts in section 3.2.2, it is necessary to understand the reasons for the occurrence of different definitions. These are elaborated in the following section 3.2.1.

3.2.1 Reasons for the Occurrence of Differences in Definitions

The various reasons that most frequently cause the occurrence of differences in definitions are examined in the following.

Reason 1: Disagreement due to Lacking Evidence

Medical knowledge is constantly growing and changing as new research results become available. Differences in definition may be due to lacking evidence which would support a given opinion. This issue is analyzed with regard to the definition of response criteria in clinical trials on Non-Hodgkin’s Lymphoma in [Cheson, Horning, et al., 1999], who collected and examined existing definitions for response assessment and conclude that “[r]esponse rates will continue to be ambiguous and subject to considerable controversy as long as we are required to base guidelines on consensus opinion of clinical data rather than on more precise and reliable measures of minimal residual disease.” (p. 1251)

Reason 2: Different Conceptualizations according to Different Purposes

In the section on the semiotic triangle (2.1), concepts were explained to be an abstraction of real-world entities. Entities in the real world can be considered to have an infinite number of properties. Every abstraction (i.e. conceptualization) of the world can represent only some subset of these properties. Thus, a unique correct conceptualization of a given entity does not exist, because it depends on purpose which properties should be represented in order to obtain a ‘good’ model. Only with some purpose in mind can one assess the importance and relevance of properties. The issue of different possible conceptualizations is widely recognized, e.g. in terminological research ([Rector, A. L., Rogers, 1999], [Lehmann, zu Bexten, 2002] p. 139) and in conceptual modeling in software development [Rumbaugh, Blaha, et al., 1993].

In analyzing the literature, we have identified three main types of differences that may occur when defining concepts, which are briefly explained below: (i) different classification, (ii) different is-a granularity, and (iii) different part-of granularity.

Different Classification

Concept definitions may be written from different points of view[14]. Depending on the point of view, different attributes are used to describe a concept and different criteria apply for classification resulting in different concept hierarchies.

Example 1:

> `Diseases may be classified from a morphological, aetiological, topological, etc., view.`

Example 2:

> A classification of the concept `<clinical trial>` according to different point of views is shown below.

point of view	subconcepts of `<clinical trial>` according to view
`organizational form`	`<multicenter study>, <monocenter study>, …`
`phase`[15]	`<phase-I-trial>, <phase-II-trial>, …`
`treated disease`	`<oncological study>, <cardiological study>, …`

Example 3:

> Different points of view may result in different attributes. Consider for example `blood`. If `blood` is considered as a `tissue`, it has a `cell structure`. However, `blood` is also a `body fluid` and thus can be described for instance by its `viscosity`.

Different Is-A Granularity

Reality can be described at different levels of detail, resulting in different levels of concept hierarchies according to the required detail of subclassification (cf. [Cimino, 1998], p. 397). The relation used in subclassification is the is-a relation (introduced in section 2.3), hence the name is-a granularity, in contrast to part-of granularity (described in the following subsection) which is based on the part-of relation.

> For example, `Insulin-Dependent Type II Diabetes Mellitus` is a concept of finer is-a granularity than `Diabetes Mellitus`. Which level of detail is adequate depends on purpose (cf. also the three- and four-character codes in the ICD described in section 4.1.1).

Different Part-Of Granularity

While the above described is-a granularity refers to different levels of is-a abstraction, concept descriptions may also vary in the amount of detail regarding the parts of a concept. The example below shows how properties of the concept `blood` are relevant at different part-of granularity levels, which correspond here to different enlargement factors.

[14] Other notions commonly used in the literature to refer to the different possible views when describing concepts are ‘perspectives’, ‘semantic axes’ and ‘semantic dimensions’.

[15] Before new drugs or treatments can be marketed, their safety and effectiveness have to be evaluated in a series of clinical trials which are classified according to the phase of this evaluation process. For further information on the characteristics of each phase see [Pocock, 1983].

blood	
enlarged by a factor of 1000	blood cells
enlarged by a factor of 10 000	cell structures
enlarged by a factor of 100 000	cell compartments visible
+ special coloring	results of processes visible

Reason 3: Interdisciplinary Character of Complex Domains

Complex domains, like the domain of clinical trials, are characterized by an interaction of experts from various fields, e.g. physicians from different subject areas, biometricians, documentation staff, computer scientists, etc. According to their different competences in certain subject fields they may be interested in different information. This is also acknowledged by [Leiner, Gaus, et al., 1999] (p. 14) who mention that different groups of persons have different information needs due to their different tasks and knowledge.

Examples:

Physicians are interested in information on diagnosis and treatment of diseases.

Biometricians are especially interested in biometrical concepts (e.g. <Karnofsky Index>), biometrical methods that may be used for the statistical analysis of certain concepts as well as information about statistical problems related to concepts.

Documentation staff are interested in any issues regarding documentation, e.g. the representation of a concept on a case report form.

Computer scientists are interested in the representation of a concept in the database.

Further, users with different competences require definitions at different degrees of difficulty. This is particularly important in the domain of clinical trials because content is to be provided for medical experts as well as for patients who are usually laymen in the domain of medicine. Laymen need different terms and a different style of definitions. Further, the kind of knowledge that is displayed may also be different: knowledge which is new and interesting for a layman is often trivial for an expert and thus should not be displayed. Different degrees of difficulty are not only important with regard to patients and non-patients, however; due to the interdisciplinary nature of the domain of clinical trials, users have competences in certain fields while they are laymen in other fields. For example, a computer scientist who is responsible for the management of the trial database does not need to have deep medical knowledge.

In addition to different interests leading to different information being relevant as described in the above examples, parts of the content may even be access-restricted and only visible for users of certain roles. By roles we mean functional roles that are related to different functions and responsibilities in the conduction of a clinical trial. As an example, the principle investigator of a study may access sensitive information that is not available to other staff.

Reason 4: Different Scientific Requirements

Specific research questions that are to be addressed by a clinical trial may also influence the definition and usage of terms. An example of such a case can be found in the documentation of clinical trials. Here, the laboratory data that are adequate for documentation may vary with respect to the aims and scope of the study. Another example for purpose-dependency is given by [Leiner, Gaus, et al., 1999]. They emphasize that deciding whether a classification or nomenclature is adequate depends on its purpose and the issues which the documentation is intended to address.

Reason 5: Different Specificity of Domain Context

The same term can be defined more generically or more specifically, depending on the domain context[16] in which the term is used. For instance, definitions in the "Guideline for Good Clinical Practice" [ICH, 1996] are written to be valid internationally and definitions in the Pschyrembel[17] are written to cover general medicine to the extent possible. In contrast, definitions in a concrete clinical trial protocol are adapted to the specific research questions and the disease being studied. For illustration purposes, an example of different definitions of `relapse` is given below.

> "Reoccurrence of a disease after remission; e.g. relapse of an infection (reinfection), tumor relapse (reoccurrence of a histologically similar tumor at the same locality or in the same organ after radical treatment)." ([Pschyrembel, Dornblüth, 1998], p. 1377, translated by the author)

> "There is relapse if, after at least 2 months CR or CRu (from the time point of the final restaging examination), one or more of the following criteria are met:
>
> - there is recurrence of disease symptoms
> - there is development of new lymphatic or extralymphatic lesions
> - there is a marked increase in lymphoma manifestation size by more than 25%.
>
> If the interval is shorter, the case is to be classified as PRO." ([Pfreundschuh, Trümper, et al., 2001], p. 50)

Reason 6: Organizational Differences

The specific particularities of an organization conducting a trial can affect definitions as well. This is especially true for SOPs since they are written to fit the needs of the organization. Another example is that different clinics which cooperate with trial-conducting centers often use different laboratory procedures and therefore supply laboratory results in special formats, use different measurement scales and different norm values.

[16] Here, 'domain context' can be understood as 'subject field'.

[17] The Pschyrembel is a lexicon of medical terms. [Pschyrembel, Dornblüth, 1998]

Reason 7: Different Geographical Location

Definitions may further vary because they are written to apply to different geographical contexts. Country-specific particularities, e.g. different laws, may influence the definition of terms. For example, a `principal investigator` (in German: Studienleiter) has different functions and responsibilities in Germany than in the USA (cf. [Heller, Lippoldt, et al., 2003a] p. 19).

Reason 8: Missing Standards

Differences in definitions may "just" be due to missing standards or missing conformance to existing standards, i.e. no substantive reason exists for the occurrence of differences (as is the case for most of the above reasons).

Many examples can be found in the area of documentation of data on CRFs, e.g.

- names of items on CRFs (e.g. `white blood count`, `WBC`, `leukocytes`, `leu.`, `leuko.`)
- different data formats (free text, multiple choice or use of codes)
- different units of measure (conventional vs. SI units).

Reason 9: Linguistic Differences

Definitions may differ because of different formulations (either within one or in different languages). Note that for natural language definitions it may be difficult to decide whether two definitions differ in formulation alone, or in content as well.

The above analysis of common reasons for differences in definitions underlines the claim of the DD (introduced in section 1.2) that developing uniform definitions is not always possible because differences in definitions may be justified. In particular, different conditions and circumstances may necessitate different definitions which cannot be harmonized (i.e. reason 4: scientific requirements, reason 5: different specificity of domain context, reason 6: organizational differences, reason 7: different geographical location). Further, different kinds of detail and different parts of knowledge may be relevant (i.e. reason 2: different conceptualizations, reason 3: interdisciplinary character of complex domains). The matter is slightly different for reason 1 (disagreement due to lacking evidence), which does not actually constitute a substantively justified difference but is simply difficult to harmonize due to the absence of conclusive research results on the basis of which a harmonized definition could be developed. However, there are also reasons which refer to unjustified differences that are possible (and desirable) to harmonize, namely reason 8 (missing standards) and reason 9 (linguistic differences[18]).

[18] Translations of the same definition to different languages are justified, of course. Different formulations within one language without differences in content are unnecessary, though, and therefore should be unified.

3.2.2 Requirements

As described in the introduction (section 1.2), one of the main objectives of the DD is to collect the different variants of definitions. From this and the abovementioned reasons for the occurrence of differences, we can derive the following requirements:

R12 Representation of Multiple Alternative Definitions

One aim of the DD is to establish a central repository of content which supports the reuse of definitions in a wide range of applications. A problem that commonly hinders the reuse of terminological systems is that they are often built for specific purposes and from specific views. However, different applications pursue different aims, have varying needs due to different views, or disagree on the definition of concepts (cf. the reasons above) and thus require different definitions.

The DD should allow this variability in the definition and usage of concepts to be represented. This contrasts with most existing terminological systems, which provide only one definition per concept.

R13 Explicit Representation of Purpose-Specific Aspects

In order to support the *correct* reuse of alternative definitions, it is necessary to explicitly represent any purpose-dependent information or underlying constraints and assumptions under which a definition is valid.

Analyzing a definition with regard to the reasons described above will help to identify purpose-specific aspects. For instance, if a definition is written to meet particular organizational circumstances (reason 6) or the legislation of some country (reason 7), it should be indicated clearly that this definition is valid and thus reusable within this particular scope only.

However, it is clear that the reasons described above are not independent of each other; for example the different conceptualizations (described in reason 2) may be of particular relevance to domain experts of certain subject fields or for accomplishing some task (reason 3).

Further, the reasons may not be clearly separable. Especially for natural language definitions, it may be difficult to assess which reason(s) of those described above cause differences in definitions. In addition, it is of course possible that more that multiple reasons occur in combination. Nevertheless, the reasons given above may help to analyze differences in definitions and to identify potential for harmonization.

In the DD model, it is therefore necessary to find a good compromise between representing as many aspects and interdependencies as possible while not overloading the model with information that is difficult to understand or with model entities that cannot be defined precisely and thus would be useless. Priority should be given to such aspects that can be represented clearly and thus will be of most benefit to the users.

R14 Context-Based Information Presentation

Reason 3 mentions the interdisciplinary character of complex domains like that of clinical trials and emphasizes the different information needs of different types of users. In order to avoid information overload, the DD should support context-based information presentation, i.e. based on the current context, it should be possible to present information selectively according to its relevance in that context.

R15 Separation of Different Hierarchical Views

As shown in reason 2, concepts can be classified on different attributes, resulting in different concept hierarchies. As pointed out by [Rector, A. L., Rogers, 1999], separating these views and providing “orthogonal taxonomies is key to re-use” (p. 4). This includes the issue of avoiding tangled hierarchies mentioned in section 2.4, namely that different relations, e.g. `is-a` and `part-of`, should not be mixed within one hierarchy, and hierarchical relations should not be mixed with non-hierarchical ones.

R16 Support for the Harmonization of Alternative Definitions

The DD should support the harmonization of the different alternative definitions. This support may be either in the form of guidelines presented to the user or constraints and rules that automatically identify potential for harmonization.

In the DD, existing alternative definitions are collected and analyzed. The aim of the harmonization process is to establish consensus where possible and necessary. Harmonization should not be made mandatory though, because the differences in definitions may be justified. Thus, the main task during harmonization is to analyze the reasons for the differences and to decide whether the differences are justified (e.g. by different purposes) or not. For unjustified differences, experts should aim towards harmonization.

The process of achieving consensus should be supported by a role-based quality assurance cycle, where only certain roles are allowed to edit and decide on consensus suggestions. Further, the discussion of consensus proposals should be supported by the DD. However, this aspect of the DD does not have high priority since discussion could happen by other means (telephone conferences, meetings, etc.). The first priority in implementation is a high quality of concept representation.

The DD could further provide guidelines on harmonization and support the automatic detection of content that should be harmonized. For instance, ambiguous terms should be detected automatically and proposed for the definition of preferred terms that are unambiguous. Further, a listing of all concepts with multiple alternative definitions could be helpful in this regard.

3.3 Functional Requirements

Note that implementing the functions below is not part of this thesis. Nevertheless, the broad functional requirements are collected for the sake of completeness as well as in order to identify additional model elements which may be required by these functionalities.

R17 The contents of the DD should be made **centrally available** (via Internet) in order to support use and reuse in various applications. The DD must provide an API[19] so that different applications can query the DD for definitions, terms, relations, etc.

R18 The DD should provide **functions to manage the content** (add, edit, delete).

R19 The DD should allow **distributed and collaborative development** of the contents. Distributed development is necessary because the members of a project (e.g. the project KML) may be based in different institutions. Collaborative development of the content is necessary because of the multidisciplinary nature of the domain. Different experts are needed to achieve high-quality definitions. From this it follows that multi-user capability is needed (i.e. concurrent access).

R20 The DD should provide a **connection to external electronically available terminological resources** (e.g. UMLS) in order to integrate these contents as alternative definitions.

R21 For applications that cannot access the DD directly (cf. R17), it should be possible to **access the content offline**, e.g. by a local copy of the DD that is regularly updated.

R22 The DD should allow to define **separate content areas**, each with access rights that can be determined locally. This allows for the management of virtually distinct DDs that are accessible by different user communities, but are all accommodated by one DD.

R23 As an extension of the previous requirement, the DD should allow for the **sharing of joint DD content areas between communities**, i.e. to allow different communities or organizations to maintain a separate content area (according to R22) for content that is access-restricted to internal members only while still being able to share other content with other communities in a joint DD content area.
For example, consider two organizations working in a similar subject area. Both could jointly edit one DD content area in which they define the concepts that are generally applicable to their work, and still each have their own DD content area where they maintain specific or confidential content. Working together in the joint DD content area is particularly desirable in view of the aim of harmonization.

[19] Application Programming Interface

R24 The DD should **allow content to be imported** (copied or moved, i.e. copied and deleted) between different DD content areas. This will be of particular interest in cases where two communities have so far developed content in separate content areas and then decide to collaborate in the development of a shared content area. In this case, it may be necessary to copy or move content from the existing separate content areas to the new joint content area.

R25 The above described function for copying content to different content areas leads to the need for **merging the content from the different content areas**, i.e. concept definitions that have been created in the separate content areas need to be merged. This may be difficult to achieve automatically, because different terms could be used in the different DD content areas to describe the same concept.

R26 The DD should provide methods to define **access restrictions to content and functions**. Access-restricted content is briefly mentioned in relation to the different roles in section 3.2.1, in reason 3 on different roles and competences resulting from the interdisciplinary character of complex domains. Access-restricted functions are needed in order to assign different rights for viewing and editing content.

R27 The DD should provide a **multilingual graphical user interface** (at least English and German) which allows users to choose their language.

4 Existing Systems

The goal of this chapter is twofold. The first aim is to provide an overview of the state of the art of terminologies by presenting some of the most well-known terminologies (section 4.1) as well as an overview of medical data dictionaries (section 4.2). The second aim is to examine these systems with regard to the goals and requirements of the DD. In particular, the analysis of existing systems focuses on two questions:

1. Representation and definition of terms and / or concepts (structure of the terminology)
2. Representation of alternative definitions, views, and contexts

The particular requirements identified for the DD with regard to these issues in sections 3.1 and 3.2 guide the answering of these questions for each terminology, but are not analyzed individually since this would exceed the space limitations of the present work. The chapter concludes with situating the DD in the state of the art in section 4.3.

4.1 Medical Terminologies

4.1.1 ICD

The *International Statistical Classification of Diseases* (ICD) is one of the most important international medical terminologies, and was first issued in 1893. Since its sixth revision in 1948, it has been maintained by the World Health Organization (WHO). The current version is the tenth revision (ICD-10) issued in 1992, the exact title of which is *International Statistical Classification of Diseases and Related Health Problems* [WHO, 1992].

The initial aim of the ICD when it was first published in 1893 was to provide an international classification of death causes in order to produce an internationally uniform and thus comparable mortality statistic. Since then, the ICD has been developed further towards documentation purposes, morbidity statistics and billing purposes (cf. [Lehmann, zu Bexten, 2002]).

Structure

Table 4.1 shows an excerpt from the ICD-10. The ICD is a monoaxial, monohierarchical classification of diagnostic terms. The hierarchical structure contains four levels of depth [Leiner, Gaus, et al., 1999]:

- 21 disease chapters (e.g. `Chapter II: Neoplasms`)
- 261 disease blocks (e.g. `C81-C96: Malignant neoplasms of lymphoid, haematopoietic and related tissue`)
- 2036 disease categories (three-character core classification) (e.g. `C81.-: Hodgkin’s disease`)
- 12161 disease subcategories (four-character classification) (e.g. `C81.1: Nodular sclerosis`)

Chapter II: Neoplasms (C00-D48) […]	
Malignant neoplasms (C00-C97) […]	
Malignant neoplasms of lymphoid, haematopoietic and related tissue (C81-C96) […]	
C81	**Hodgkin’s disease** ***Includes:*** morphology codes M965-M966 with behaviour code /3
C81.0	**Lymphocytic predominance** Lymphocytic-histiocytic predominance
C81.1	**Nodular sclerosis**
C81.2	**Mixed cellularity**
[...]	
C82	**Follicular [nodular] non-Hodgkin's lymphoma** ***Includes:*** follicular non-Hodgkin's lymphoma with or without diffuse areas morphology code M969 with behaviour code /3
C82.0	**Small cleaved cell, follicular**
C82.1	**Mixed small cleaved and large cell, follicular**
C82.2	**Large cell, follicular**
C82.7	**Other types of follicular non-Hodgkin's lymphoma**
C82.9	**Follicular non-Hodgkin's lymphoma, unspecified** Nodular non-Hodgkin's lymphoma NOS
C83	**Diffuse non-Hodgkin's lymphoma** ***Includes:*** morphology codes M9593, M9595, M967-M968 with behaviour code /3
C83.0	**Small cell (diffuse)**
C83.1	**Small cleaved cell (diffuse)**
C83.2	**Mixed small and large cell (diffuse)**
C83.3	**Large cell (diffuse)** Reticulum cell sarcoma
[...]	

Table 4.1: Excerpt from ICD-10[20].

The ICD does not contain definitions of the diagnoses. The meaning of a diagnostic term is thus described only by its place in the hierarchy and the inclusion and exclusion comments[21]. The only exception is chapter V which covers mental and behavioral disorders. Diagnoses in this chapter are accompanied by a glossary because it is acknowledged that particular difficulties arise in this subject field due to internationally unstandardized terminology – many common terms are used with different meanings.

For some diagnoses, the ICD provides a limited possibility to define complex concepts by using two codes in conjunction. This is called the dagger/asterisk-system (+/*), where the first code (the +-code) refers to the cause of the disease (its aetiology) and the second code (the *-code) specifies the localized manifestation of the disease. Apart from this system and the hierarchical relations between concepts, relations are lacking in the ICD.

Regarding the multilingual accessibility of the ICD, there exist translations in many languages which are maintained by organizations other than the WHO, e.g. the German Institute of

[20] source: http://www.dimdi.de, online version of ICD-10, version for 2003

[21] The exclusion and inclusion comments ensure that the classes in the ICD are disjoint, which is essential for classification as mentioned in section 2.2.

Medical Documentation and Information (DIMDI) is responsible for the German translation. Synonyms are supported by means of an alphabetical index which is part of the ICD.

Alternative Definitions, Views and Contexts

Alternative definitions are not provided by the ICD, which is understandable because the aim is to establish an internationally uniform statistical classification. Thus, consensus is absolutely necessary – maintaining alternative definitions would not be useful for this purpose. However, it is worth mentioning here that the lack of precise definitions in the ICD is problematic. As mentioned above, the only descriptive information about a concept (i.e. a diagnostic term) is its place in the hierarchy as well as the inclusion and exclusion comments. This is surely not enough and poses a problem with regard to the reproducibility of classification because the delineation of similar clinical pictures is not well-defined and the common usage of diagnostic terms may vary internationally. A glossary with at least informal definitions (as contained in chapter V as explained above) would be one step towards greater precision.

Furthermore, the ICD is an example of a terminology which is very purpose-specific. This concerns the detail of classification as well as the organization of terms in the hierarchy. As pointed out by [Lehmann, zu Bexten, 2002], the definition of classes in the ICD is mainly motivated by statistical concerns, e.g. the prevalence of diseases. As a result, rare diseases are classified in less detail than diseases which occur more frequently. The hierarchy is purpose-specific insofar as different views are mixed: the semantic perspective which is used for classification changes between topography, pathology, and aetiology.

The ICD can be considered as providing some support for views with respect to different granularities because one can either use the three- and four-character codes for coding, depending on what level of detail is required for a given purpose. In Germany, for example, only medical specialists within their own fields are obliged to use the four-character codes while general practitioners may use the three-character codes. As another example, according to [van Bemmel, Musen, 1997] (p. 89), three-character codes are also sufficient for reporting mortality statistics to the WHO.

Apart from this, the ICD does not provide any support for different views or contexts. In particular, as a monoaxial classification the ICD does not allow classification of diseases along different points of view (as described in reason 2 in section 3.2.1).

4.1.2 SNOMED

The *Systematized Nomenclature of Medicine* (SNOMED®) [SNOMED International, 2004] is a nomenclature for human and veterinary medicine which can be used for example to index information in the patient record (e.g. diagnoses, signs and symptoms, findings, observations, procedures, etc.), lab reporting, information retrieval, etc.

SNOMED is developed and maintained by SNOMED International, a division of the College of American Pathologists in the United States[22] [SNOMED International, 2004]. SNOMED has evolved over the past forty years through several editions (1977: SNOMED, 1979: SNOMED II, 1993: SNOMED III, 2000: SNOMED Reference Terminology (SNOMED RT)). The latest

[22] http://www.cap.org

edition was released in 2002, is called *SNOMED Clinical Terms (SNOMED CT)* and integrates the NHS Clinical Terms[23] into SNOMED. SNOMED RT contains more than 120.000 concepts [Stearns, Price, et al., 2001], SNOMED CT over 357.000 concepts [SNOMED International, 2004].

Structure

The structure of SNOMED has evolved significantly over its several editions. In particular, there have been major changes between SNOMED III and SNOMED RT reflecting the advancement in medical terminology research [Dolin, Spackman, et al., 2001; Spackman, K.A., 1999]. Hence, we compare these two versions in the following brief introduction to SNOMED in order to illustrate the changes due to progress in terminological research. (We do not cover SNOMED CT because changes in this edition primarily relate to the integration of the NHS Clinical Terms.)

SNOMED is a multiaxial nomenclature. The SNOMED III edition consists of eleven axes (also called modules, chapters or dimensions), each forming a separate hierarchical classification: Topography (anatomy – abbreviation T); Morphology (pathologic structure – M); Living Organisms (Bacteria and viruses – L); Chemicals (Drugs and Biological Products – C); Function (Signs and Symptoms – F); Occupation (job – J); Diseases, Diagnoses (D); Procedures (P); Physical Agents, Activities and Forces (A); Social Context (S); General (syntactic linkages and qualifiers – G) (cf. [Leiner, Gaus, et al., 1999; van Bemmel, Musen, 1997]).

The concepts from these axes can be combined in order to form a complex concept expressing a clinical statement. Some concepts cannot, however, be combined arbitrarily with concepts from other axes but contain references to concepts of other axes. In particular, many disease concepts cross-reference concepts in other axes which are essential characteristics of the disease [Spackman, Kent A., Campbell, et al., 1997]. An example of a diagnosis which is defined by reference to other axes is given in the following (according to [Spackman, Kent A., Campbell, et al., 1997] p. 642).

SNOMED III termcode		SNOMED III components of the concept (cross-reference fields)				
D	=	**T**	+	**M**	+	**F**
Postoperative esophagitis	=	Esophagus	+	Inflammation	+	Post-operative state
D5-30150	=	T-56000	+	M-40000	+	F-06030

Complex concepts can be further qualified, e.g. the qualifier `FH` can be used in front of a code in order to express that the disease is part of the patient’s family history (this is related to the notion of context in SNOMED; see below). Furthermore, it is possible to combine multiple clinical statements (expressed by a complex concept) by syntactical links, e.g. `associated with`, `as a result of`, etc. ([Leiner, Gaus, et al., 1999], p. 60).

[23] The *NHS Clinical Terms* is a terminology which was developed by the National Health Service in the United Kingdom. It was formerly known as READ Codes. Further information: http://www.nhsia.nhs.uk/terms/

In SNOMED RT, major changes have been made in order to apply "the principles of good terminology practice" and include ([Spackman, K.A., 1999], p. 24):

- explicit multiple hierarchies,
- use of description logic to define concepts.

The latter of these changes requires some more explanation. Like every terminology which allows for the definition of complex concepts, SNOMED faces the problems of redundancy (multiple ways to express the same concept) as well as the possibility to create nonsense concepts. Rules for the combination of complex concepts which could avoid these issues are lacking in SNOMED and are difficult to develop. For this reason, the developers of SNOMED decided to use description logic to define concepts in SNOMED RT, which allows for the automatic detection of different descriptions that are semantically equivalent and thus helps to avoid redundancy. Further, multiple hierarchies can be computed automatically (or inferred) using formal definitions written in description logic, which provides significant support in maintaining the consistency of the terminology.

The following example illustrates the representation of `Postoperative esophagitis` in SNOMED RT [Spackman, K. A., 2001]:

```
D5-30150:
        D5-30100 &
                (assoc-topography T-56000) &
                (assoc-morphology M-40000) &
                (assoc-etiology F-06030)
```

Each concept definition is a single logical expression containing the supertypes of the concept being defined (in this example, the concept identified by the code `D5-30100`), as well as so-called attribute-value pairs (the latter three lines in the example) describing the essential characteristics of the concept.[24] A more detailed explanation of description logic is beyond the scope of this thesis. Further information regarding description logic in SNOMED can be found in [Spackman, K. A., 2001; Spackman, Kent A., Campbell, et al., 1997]. A general and comprehensive introduction to description logic is provided by [Baader, Calvanese, et al., 2003].

Alternative Definitions, Views and Contexts

SNOMED does not provide alternative definitions. As described in [Dolin, Spackman, et al., 2001], concept definitions are developed based on an analysis of existing terminologies and standards. They acknowledge that the definitions provided by these standards are often conflicting, but state that SNOMED develops a "single best definition" ([Dolin, Spackman, et al., 2001], p. 140). Furthermore, [Dolin, Spackman, et al., 2001] (p. 141) mention that sometimes there are very subtle differences in the meanings of two concepts, for example the procedures `insertion` vs. `implantation`. Such fine differences are not represented in SNOMED but rather modeled as one concept with synonyms.

[24] In description logics, attribute-value pairs are usually preceded by a quantifier, i.e. either the existential quantifier ($\exists$) or the universal quantifier ($\forall$). The developers of SNOMED RT have chosen to use only the existential quantifier and thus, for easier readability, this quantifier is implicit in the syntax of SNOMED attribute-value pairs.

Regarding the precision of definitions, however, it can be observed that SNOMED RT has aimed at making definitions more precise by making implicit characteristics explicit through additional attributes and / or multiple superconcepts ([Dolin, Spackman, et al., 2001], p. 141). Thus, ambiguity of definitions is reduced and higher reproducibility is achieved in SNOMED RT (reproducibility means that the same medical fact should not be coded differently by different persons).

An advantage of SNOMED RT is that it separates multiple hierarchical views more cleanly than previous editions of SNOMED and many other terminologies. The type of hierarchy is also made explicit (e.g. `is-a`, `part-of`, etc.) and definitions written in description logic support the consistency of the multiple hierarchies [Spackman, Kent A., Campbell, et al., 1997]. Thus, SNOMED RT fulfills the requirement of separating different hierarchical views as asked for by requirement R15.

The qualifiers mentioned above represent what can be considered as contexts in SNOMED. For example, the concept of `myocardial infarction` is a heart disease which, when used in a patient record, is interpreted as a disease in the patient's history. However, if qualified by `FH`, it becomes a `family history of myocardial infarction` representing a disease in the patient's family history. Thus, the context in which the concept of `myocardial infarction` is used changes its interpretation.

4.1.3 LOINC

LOINC® is an acronym which stands for *Logical Observation Identifier Names and Codes,* and "provides a universal code system for reporting laboratory and other clinical observations" ([McDonald, Huff, et al., 2003], p. 624). It has been under development since 1994 in a voluntary effort initiated and coordinated by the Regenstrief Institute for Health Care in Indianapolis (USA). The current release, version 2.12 (released in February 2004), contains more than 34.000 codes [LOINC Committee, 2004a].

The development of LOINC is motivated by the same issues as any standardized terminology: different IT systems communicate and exchange data, thus standardization is necessary in order to allow for automated data processing as well as integration of data from different sources. In the case of the LOINC project, the scenario is that laboratory systems send their results electronically to the receiving systems (e.g. hospitals, research groups). The syntax of such messages has already been standardized by the Health Level Seven standard (HL7)[25], but the latter has not standardized which codes or terms are to be used for identifying certain laboratory tests. Hence, without LOINC, laboratories send their own, unstandardized codes in HL7 messages. Consequently, the same test is coded differently by different laboratories, and the receiving hospital needs to map the different codes in order to integrate the data.

The purpose of LOINC is therefore to provide a universal coding system which can be used in existing messaging standards like HL7 for identifying laboratory and other clinical observations, e.g. vital signs like blood pressure, temperature, pulse rate [McDonald, Huff, et al., 2003]. LOINC does not standardize the result values of tests – other terminologies and code systems are used for this purpose (e.g. ICD, SNOMED).

[25] http://www.hl7.org

Structure

The structure of LOINC is described below according to the LOINC users' guide [LOINC Committee, 2004a].

LOINC is a multiaxial nomenclature with the following six axes:

1. *Analyte / component* – e.g. potassium, hemoglobin, hepatitis C antigen, systolic blood pressure
2. *Property measured* – e.g. a mass concentration, enzyme activity (catalytic rate), pressure
3. *Time aspect* – i.e. whether the measurement is an observation at a moment of time, or an observation integrated over an extended duration of time – e.g. point in time, one hour, 24 hours
4. *System / sample* – e.g. urine, blood, patient
5. *Type of Scale* – e.g. quantitative, ordinal, nominal, narrative
6. *Method* (only where relevant) – i.e. the method used to produce the result or other observation

Each test (e.g. a leukocyte count, or a body temperature measurement) is a complex concept which is defined by choosing an elementary concept from each of the six axes (except for the method axis which is optional). The meaning of the elementary concepts which are used as values for each axis is explained in the users' guide. Table 4.2 lists some examples of LOINC codes.

LOINC code	analyte/ component	property measured	time aspect	system/ sample	scale	method
8312-1	BODY TEMPERATURE	TEMP	8H^MAX	XXX	QN	
11289-6	BODY TEMPERATURE	TEMP	ENCTR^FRST	^PATIENT	QN	
26464-8	LEUKOCYTES	NCNC	PT	BLD	QN	
6690-2	LEUKOCYTES	NCNC	PT	BLD	QN	AUTOMATED COUNT
804-5	LEUKOCYTES	NCNC	PT	BLD	QN	MANUAL COUNT
1994-3	CALCIUM.IONIZED	SCNC	PT	BLD	QN	

Table 4.2: Examples of LOINC codes [LOINC Committee, 2004b].

LOINC is a precoordinated nomenclature, i.e. all allowed combinations of values from the six axes are predefined and assigned a unique code. In this sense, it is also a classification because each observation is identified by exactly one code.

The danger of combinatorial explosion, which is inherent to terminologies enumerating the possible combinations of values from different axes, is circumvented in LOINC by two practices [Huff, Rocha, et al., 1998]. Firstly, the strategy is to define only codes which are actually required by real systems, and secondly, in order to reduce complexity and the number of possible combinations, other fields in HL7 messages are used for representing information not included in the LOINC code, e.g. "the instrument used in testing" or "the size of the sample collected" ([LOINC Committee, 2004a], p. 15).

As can be seen from the examples given in table 4.2, the values for each of the axes are not always atomic but carry some internal structure themselves. Two special characters are used: the caret (`^`) and the dot (`.`).

The caret is used for separating subparts of an axis. The time axis, for example, is divided into two subparts, where the first describes the time aspect (e.g. `PT` for point in time, or `8H` for a duration of eight hours) and the second part is an optional modifier used to indicate "some subselection or integration of the measures taken over the defined period of time" ([LOINC Committee, 2004a], p. 31). In the first example of the body temperature given in table 4.2, the value of the time aspect modifier is `MAX` in order to indicate that the value reported is the maximum value measured in the time interval of `8H` (= eight hours).

The dot is used in the first subpart of the first axis (analyte / component) for separating multiple levels of increasing taxonomic specification. An example is `CALCIUM` which is one component itself, and `CALCIUM.IONIZED` which is also a component and a subclass of `CALCIUM`. Both can be used as values in the first subpart of the first axis.

Furthermore, LOINC also contains some hierarchical structure insofar as the codes are categorized broadly into four *class types* (`laboratory`, `clinical class`, `claims attachments`, `surveys`) which are each further divided into *classes* (e.g. examples of classes included in the class type `laboratory` are `chemistry`, `cytology`, `microbiology`, `molecular pathology`, etc.). As emphasized in [LOINC Committee, 2004a], this classification is to some extent arbitrary and is included in order "to make it easier to find general areas of interest" (p. 9). Users of LOINC are expected and encouraged to add their own useful classification information to the database.

The linguistic level, i.e. the terms and synonyms that can be used to refer to LOINC concepts, is not represented in a detailed manner in LOINC. Although words of natural language are contained in LOINC for each concept, these words are rather synonyms, abbreviations and related terms of the elementary concepts that a complex concept consists of, instead of terms which can be used to refer to the whole complex concept. For example, all three tests in table 4.2 which determine the leukocyte count (i.e. with the value `LEUKOCYTES` in the first axis), have among others the synonyms `white blood cells`, `wbc`, `leuc`, `leuk`. The complex concept itself can be referenced unambiguously (apart from using the code) by using the "fully specified LOINC name" or "long formal name". This name is created by appending the values from the six axes, separated by a colon [LOINC Committee, 2004a]. The long formal name of the first example in table 4.2 is thus `BODY TEMPERATURE:TEMP:8H^MAX:XXX:QN`. From this example it is obvious that the name is formally generated instead of being taken from natural language. In view of the huge number of tests coded in LOINC this is reasonable because choosing a natural language expression for each test is not feasible due to the high level of detail in concept representation.

Alternative Definitions, Views and Contexts

LOINC neither supports alternative definitions nor context-specific information for a concept. A particular deficiency of LOINC is its lack of different granularities – the high level of detail in LOINC without support for hiding irrelevant detail is an issue which complicates its usage.

Furthermore, LOINC exhibits an inhomogeneous level of granularity. This is due to the pragmatic approach chosen regarding the decision of what should be represented as one or as different concepts: a difference between two similar tests is represented in LOINC by two codes instead of only one if it is of practical relevance, i.e. if it is used by some laboratory in practice. This is particularly the case for the method axis, since for many chemical and hematological tests the method is often not significant while it is significant for immunochemical tests because of high variances in sensitivity and specificity (cf. [LOINC Committee, 2004a], p. 36). The rationale for this approach is, as mentioned above, to avoid unnecessary complexity and combinatorial explosion of the number of concepts.

4.1.4 UMLS

The *Unified Medical Language System* (UMLS®) is a project being developed at the U.S. National Library of Medicine (NLM) since 1986.[26] The development of the UMLS was motivated by the increasing number of heterogeneous terminological systems which represent, classify and name the same concepts differently, and thus provide a barrier to retrieval and integration of information from different sources. The goal of UMLS is to integrate multiple machine-readable biomedical information sources (e.g. ICD, LOINC, SNOMED, etc.) as transparently as possible. The information given in this section refers to the UMLS in its current 14th edition [NLM, 2003].

Structure

The UMLS consists of three parts – the three *UMLS Knowledge Sources*:

1. the Metathesaurus,
2. the Semantic Network,
3. the SPECIALIST lexicon and other related lexical programs.

The *Metathesaurus* is the component which ties together the different source vocabularies. A concept in the Metathesaurus links together concepts from different source vocabularies which are judged by the UMLS editors to have the same meaning. An important principle of the Metathesaurus is that the information of the underlying source vocabularies (terms, concepts, codes, attributes, hierarchical and other relations) is preserved in order to achieve a transparent integration (hence it is called “Meta-”thesaurus), and synonymy information and other relations between terms and concepts from different vocabularies are added by the UMLS editors in order to achieve integration ([NLM, 2003], p. 12). More than 970.000 concepts are currently contained in the Metathesaurus ([NLM, 2003], p. 7).

The *Semantic Network* provides a categorization of the concepts contained in the Metathesaurus. The current version of UMLS defines 135 semantic types, which are organized hierarchically (by the is-a relationship) into two distinct hierarchies, rooted by `Entity` and `Event` respectively. Each concept is assigned to at least one semantic type (but only the most specific ones available). A portion of the hierarchical structure of the Semantic Network is shown in table 4.3.

[26] http://www.nlm.nih.gov/research/umls/umlsmain.html

```
Entity                                      Event
  Physical Object                             Activity
    Organism                                    Behaviour
      Plant                                       Social Behaviour
        Alga                                      Individual Behaviour
      Fungus                                    […]
      Virus                                   Phenomenon or Process
      […]                                       […]
    Anatomical Structure                        Natural Phenomenon or Process
      […]                                         Biologic Function
      Fully Formed Anatomical Structure             Physiologic Function
        […]                                           […]
        Tissue                                      Pathologic Function
        Cell                                          Disease or Syndrome
        Cell Component                                […]
  […]                                         […]
```

Table 4.3: Excerpt from the Semantic Network hierarchy [NLM, 2003]. Subtypes are indented below their supertypes.

The developers of UMLS acknowledge that the Semantic Network exhibits a varying level of granularity which is justified by the aim of avoiding too many semantic types ([NLM, 2003], p. 46). In particular, this means that for many semantic types, their subcategories are not exhaustive but rather only the ones most important to the biomedical domain are defined, and concepts belonging to the subcategories not defined in the Semantic Network are assigned to the more general semantic type. As an example, the semantic type `Alga` is the only subcategory of `Plant`, although there are obviously many other conceivable subcategories. Apart from the hierarchical relation, there also exist non-hierarchical relations between semantic types, which are described below.

The third component of the UMLS knowledge sources, the *SPECIALIST lexicon*, is included in the UMLS in order to support linguistic applications. It contains for each entry “syntactic, morphological, and orthographic information” ([NLM, 2003], p. 53). The included lexical programs support UMLS users in abstracting from lexical variants of words, like inflected forms (e.g. singular / plural of nouns, or past / present tense of verbs) and the word order variants of multi-word terms.

The UMLS is a concept-oriented system with the special characteristic that it defines three instead of two levels, i.e. it distinguishes between *concepts*, *terms* and *strings*, while most systems distinguish between concepts and terms only. Each possible designation of a concept is stored as a string in UMLS with a “String Unique Identifier” (SUI). In order to group strings which are lexical variants of each other (e.g. plural and singular variants, inflected word forms, or spelling and case variants), the intermediate level of terms is additionally introduced in the Metathesaurus. Terms are identified by a “Lexical Unique Identifier” (LUI). The UMLS is multilingual insofar as each string has an attribute for the language of the string.

In UMLS, two types of relations are distinguished: i) relations between semantic types in the Semantic Network and ii) relations between Metathesaurus concepts (inter-concept relations). Both types of relations are explained below. Note that all relations are binary, i.e. they relate two concepts.

As mentioned briefly in the description of the Semantic Network above, relations between semantic types can be further divided into the hierarchical relation is-a and non-hierarchical relations (e.g. process of, evaluation of, causes, produces). The is-a relation establishes an inheritance hierarchy of semantic types[27], and the non-hierarchical relations are defined between the most general semantic types possible and inherited along the is-a hierarchy to all subtypes. However, inherited relations can be blocked for subtypes if they do not apply there ([NLM, 2003], p. 47). The current version of UMLS contains 54 semantic relations.

The meaning of non-hierarchical relations stated between semantic types is that they describe potential inter-concept relations between the concepts categorized under the related semantic types, i.e. a relation between two semantic types may or may not occur as an inter-concept relation between any particular pair of concepts assigned to these semantic types ([NLM, 2003], p. 46). See figure 4.1 for an example. The relationship evaluation of holds between overweight and body weight, but not between fever and body weight. In terms of UML [OMG, 2003a], a relation between two semantic types corresponds to an association in UML, and the inter-concept relations to links which instantiate the association.

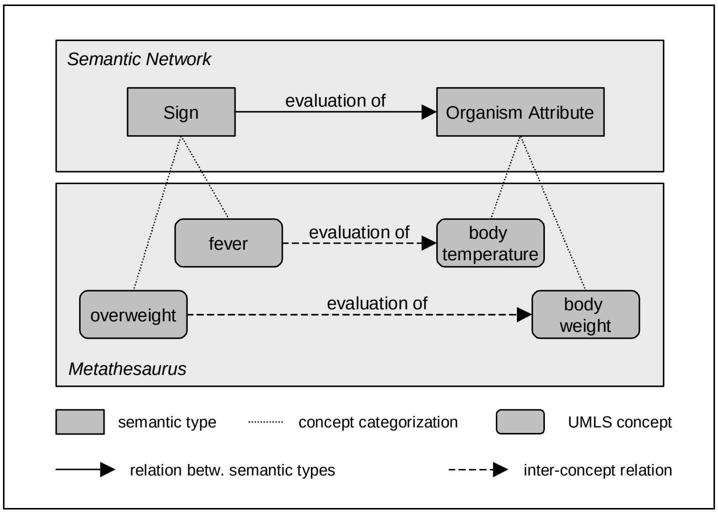


Figure 4.1: Examples of relations in the UMLS.

Inter-concept relations in the Metathesaurus are described by two attributes. One attribute characterizes the type of the relation, which is rather unspecific[28]. A more specific characterization of the nature of relations can be achieved by the second attribute, which can take as its value, among others, a relation defined in the Semantic Network, as is the case in the example of the inter-concept relation evaluation of in figure 4.1 above. However, only a small percentage of inter-concept relations has a value for this attribute ([NLM, 2003], p. 112).

All relations in the UMLS are directed, and two relations can be defined as being inverses of each other. An example of relations which are defined as being inverses of each other are affects and affected_by ([NLM, 2003], p. 52).

[27] Note that although concepts in the Metathesaurus are also hierarchically related, this hierarchy is not used for inheritance because the different source vocabularies define very different hierarchical structures which are all included in the Metathesaurus. Hence, the concept hierarchy in the UMLS may contain cycles and is thus unsuitable for inheritance (cf. [Bodenreider, 2001] p. 4).

[28] There are eleven types of relationships in the Metathesaurus, e.g. narrower, broader, or other related. For more information see [NLM, 2003], p. 15.

Alternative Definitions, Views and Contexts

To summarize, concepts are defined in UMLS by means of their associated terms and strings (synonyms), their semantic type(s), and by relations to other concepts. If a vocabulary contains a textual definition of the concept, it is also included in the Metathesaurus. Hence, a concept can contain multiple textual definitions.

The UMLS is, to our knowledge, the only terminological system which contains concept definitions from multiple sources, and can hence be regarded as contributing one step towards alternative definitions of concepts. The advantage of the UMLS is that it achieves the integration of the various source vocabularies in a transparent manner by always maintaining a reference to the source of some information (e.g. each relation which is derived from some source vocabulary is marked as such).

Nevertheless, the UMLS has shortcomings with respect to the requirements for the representation of alternative definitions described in chapter 3. Firstly, most definitions which are included from the source vocabularies lack precision and fail to indicate the constraints on reuse (e.g. geographical constraints) as asked for by requirement R13. Relations are also often characterized imprecisely (cf. the description of relations above: most inter-concept relations only specify the type of the relationship but do not carry a value describing the exact nature of the relationship). This lack of precision hampers a comparative analysis of alternative definitions as well as their correct reuse.

Views, in the sense of a subset of content of the UMLS, can be created using a special tool called *MetamorphoSys* ([NLM, 2003], p. 40). This software allows for the selection of certain source vocabularies from the UMLS and the exclusion of others in order to adapt the UMLS to the user's purpose. Excluding source vocabularies may be necessary due to license restrictions which are different for each source. In addition to including whole source vocabularies, there are also filters which allow for the exclusion of specific attributes, languages, relationship types and semantic types. It is also possible to define custom filters.

The notion of context is defined in UMLS as the place "of a concept in a hierarchy in any of the UMLS source vocabularies", i.e. ancestor, sibling, or child concepts, ([NLM, 2003], p. 31). Most concepts have multiple contexts, either from occurrences in multiple source vocabularies or due to multiple hierarchical positions in one source vocabulary. The purpose of contexts in the UMLS is to aid the users in "understanding the scope of concepts" ([NLM, 2003], p. 105).

4.1.5 GALEN

Among currently existing terminological systems, GALEN [OpenGALEN, 2004b] is one of the most sophisticated and comprehensive ones. The acronym GALEN stands for *Generalised Architecture for Languages, Encyclopaedias and Nomenclatures in Medicine*. As the name suggests, it is more than just a terminology and provides a whole framework for medical terminology, including a conceptual model of the medical domain and supporting software.

The development of GALEN is motivated by the fact that most traditional terminologies (e.g. ICD) are developed for specific purposes, which becomes evident, for example, in hierarchies organized according to specific organization principles supporting the terminology's intended usage, in the axes chosen in multiaxial systems, and in a certain level of detail in concept representation. These issues are typical hindrances to the reuse of existing terminologies for different purposes requiring different principles of organization and levels of detail.

The goal of GALEN is to overcome these problems by providing an application and purpose-independent framework for clinical terminology. The range of application areas it is envisioned for is very broad and includes, among others, medical records, clinical information systems, decision support systems, natural language understanding systems, mapping between different coding systems, and information retrieval (cf. [Rector, A. L., Solomon, et al., 1995], p. 149).

GALEN has been developed from the beginning of the nineties by the GALEN consortium, coordinated by Rector et al. (Medical Informatics Group at the University of Manchester, U.K.) [Rector, A. L. , Nowlan, et al., 1993] and was funded by the European Community until 1995. The non-profit organization OpenGALEN is now distributing the GALEN Common Reference Model (see below) under an open-source license [OpenGALEN, 2004b].

Structure

The three main components of GALEN are:

1. the GALEN *Common Reference Model*,
2. the representation language *GRAIL* (GALEN Representation And Integration Language),
3. the GALEN *Terminology Server*.

The Common Reference Model contains the medical concepts which are represented in the formal language GRAIL. GRAIL is a complex and expressive language which is similar to description logic languages, and thus provides automatic multiple classification (i.e. multiple hierarchies can be inferred from the formal concept definition). It was developed specifically for addressing the complex needs of medical terminology. Finally, the Terminology Server is the software which implements GRAIL and provides several services, e.g. accessing and querying the Common Reference Model, as well as mapping to external terminologies.

According to [Rogers, J. E., Rector, 1999], the Common Reference Model is structured into the following four layers (however, the dividing lines between these layers are not sharp):

1. the high level ontology (very basic concepts and broad patterns for composing more detailed concepts),
2. the Common Reference Model itself (reusable concepts of anatomy, diseases, clinical signs, etc., and constraints for combining them),
3. detailed extensions to the building blocks of the previous two levels (e.g. more information on body regions, types of surgery),
4. models of specific domains which define complex concepts on the basis of the upper levels (e.g. the model of surgical procedures).

While the second level is expected to be reusable by almost all applications, levels three and four are more specific to certain applications and subdomains.

The high level distinction in the GALEN Common Reference Model is between *things* and *properties*. The first is divided into the categories `GeneralisedStructures`, `GeneralisedSubstances`, `GeneralisedProcesses`. The properties of things are categorized under

`ModifierConcept`, which is subdivided into `Aspect` (e.g. `location`, `Shape`)[29], `Role` (e.g. `PatientRole` or `DoctorRole`), `Modality` (e.g. `FamilyHistory`, `Risk`), `Collection` (e.g. `set`), and `Unit` (e.g. `metre`, `second`).

The categories described above can be linked together by so-called *attributes*. Attributes are organized hierarchically in GALEN, defining the main distinction between `Constructive-Attribute` (e.g. `connects`, `hasLocation`), `ModifierAttribute` (`hasFeature`, `hasRole`), and `TemporalAttribute` (not developed in detail in GALEN, but reserved for relationships like `occursDuring`) (cf. [Rogers, J. E., Rector, 1999], p. 12).

GALEN makes extensive use of compositional concept definition and GRAIL was developed with the aim of providing as much automatic support as possible. It is interesting to note that in order to be able to combine two concepts, this combination must explicitly be allowed by some statement in the Common Reference Model. This mechanism is known as *sanctions* in GALEN and avoids the definition of complex concepts which are medical nonsense (which can happen, for example, in SNOMED). GRAIL also helps to deal with the problem of redundancy because it can determine automatically whether two concept definitions are semantically equivalent.

Finally, two examples of concept definitions in GRAIL are given below. The first example is the definition of the concept of disease, while the second defines a subtype of disease, namely a heart disease.

```
Phenomenon which hasPathologicalStatus pathological.

(Phenomenon whichG <
      LocativeAttribute Heart
      hasPathologicalStatus pathological>) name HeartDisease.
```

This brief introduction to the Common Reference Model should convey the significantly greater complexity of GALEN, as compared to the models of any of the previously introduced terminologies.

Alternative Definitions, Views and Contexts

As stated in [OpenGALEN, 2004a], GALEN represents only consensus knowledge. In particular, [Rector, A. L., Rogers, 1999] emphasize that definitions sometimes vary among authorities and that representing such differences is beyond the scope of GALEN. From this it can be concluded that alternative definitions resulting from different expert opinions are not represented in GALEN.

However, as described in the reasons for the occurrence of alternative definitions (section 3.2.1), possible reasons include different conceptualization due to different purposes and views. GALEN provides support in this respect due its purpose-independent concept representation and the multiple hierarchies, which can automatically classify the same concept according to different views (which fulfills our requirement R15: Separation of Different Hierarchical Views). Further, GALEN allows for the application of a “set of filters and systematic simplifications” to the Common Reference Model resulting in a “reduced view of the Common

[29] The examples of categories given in parentheses are actually subtypes of the respective categories, occurring low or as leaves in the hierarchy.

Reference Model being visible to the outside" ([Rogers, J., Roberts, et al., 2001] p. 259). In other words, this mechanism allows for the creation of purpose-specific subsets of the Common Reference Model, e.g. in order to exclude irrelevant details.

GALEN does not define a specific notion of context. Due to the high expressiveness and flexibility of GRAIL, however, it is capable of representing many aspects which are typically lacking in traditional terminologies and usually related to the notion of context. In particular, this concerns the availability of multiaxial classification as well as the categories `Role` and `Modality`. An example is the issue of a disease occurring as a present diagnosis of the patient or in the patient's family history. These two meanings can be represented using the modality `FamilyHistory` (cf. [Rogers, J. E., Rector, 1999]).

4.2 Medical Data Dictionaries

In this section, an overview of medical data dictionaries is provided. In contrast to the previous section in which well-known medical terminologies were presented individually, we follow a different route for medical data dictionaries here. Instead, we will summarize the main characteristics which are common to these systems and relate these to the goals and requirements of the DD. For a current and comprehensive overview of individual systems we refer the reader to [Bürkle, 2000].

First of all, according to the definition of medical data dictionary (MDD) given in 2.1, an MDD contains the local terminology of some institution, usually a hospital. In contrast to terminologies such as those presented in the previous section, an MDD may provide further functionalities for supporting the local hospital information system. After a detailed review of existing systems, Bürkle summarizes the potential application areas of MDDs as follows (cf. [Bürkle, 2000], p. 130):

- support of structured documentation and data storage,
- translation of local vocabulary to external classifications and codes,
- integration of systems by providing terminological translation,
- knowledge-based functions (medical decision support, generation of warnings),
- web-based information retrieval,
- context-sensitive presentation of information, and
- text analysis, text production and translation of texts.

The structure, complexity, and architecture of an MDD vary greatly between different implemented systems. Bürkle classifies MDDs into three generations which are summarized below.

The first generation of MDDs was primarily motivated by restricted storage capacity and tried to avoid saving redundant information or long result texts (diagnoses, observations, etc.). Therefore, tables were designed containing the unique mapping from codes to the corresponding concepts (e.g. a diagnosis). The documentation in the patient record thus needed to store only the short codes referencing the concept in the DD. In order to structure these code lists, concepts were organized into hierarchies and categories according to different medical fields of work. Such groups of concepts were also used for statistical analyses as well as for other purposes.

Further, the problem of synonyms was recognized: it was important to avoid the possibility of assigning two different codes to the same medical fact because this would prevent the reliable retrieval of information. The first generation of data dictionaries can thus be summarized as supporting standardized documentation of patient data by providing a unified vocabulary together with synonyms. First generation MDDs were tightly integrated into the information system.

The second generation identified by Bürkle was driven by technological progress. As storage capacity and speed of computer systems increased rapidly and relational databases were developed, the technical restrictions became less important. Thus, model restrictions with regard to the number of concepts and relations as well as the size or the levels of the hierarchy that could be represented in an MDD were abolished. The development of MDDs could now primarily be oriented towards functional instead of technical requirements.

The development of the third generation is characterized by two main properties. Firstly, these data dictionaries were developed as independent systems – so-called data dictionary servers. A data dictionary server provides an interface through which other information systems can send queries. While MDDs of previous generations were tightly integrated with the hospital information system, the independence of a data dictionary server from other systems allows for the reuse of the DD and its integration with different information systems. Secondly, increasing requirements with regard to the vocabulary to be represented (especially with regard to its size) led to the development of more advanced concept representations. Compositional approaches that defined complex concepts on the basis of elementary concepts were developed. The latter issue in particular is still a research topic, however, and is hardly available in practice.

As with terminologies in general, there exists no unified method or representation language for constructing MDDs. Existing systems have all developed their own proprietary structure for representing concepts, most of which use means similar to the five terminologies described in section 4.1 for representing concepts, e.g. concept hierarchies and relations to other concepts. Natural language definitions of concepts are hardly contained in MDDs. Further, formal definitions (e.g. in a description logic as in GALEN and SNOMED RT) and complex concept representation are also uncommon (cf. [Bürkle, 2000]).

The major difference between terminologies as presented in the previous section and MDDs is that the latter also represent concepts and relations which are specific to local applications. As an example, the Medical Entities Dictionary (MED)[30] defines a relation `is-display-parameter-of` which relates a concept to other, rather technical concepts like `Urinalysis-Button` (which apparently refers to a button in an application whose functionality depends on the concepts and their relations in the MED).

To conclude we address an issue which is particularly interesting for this work relating to the application areas of MDDs mentioned above, namely that of context-sensitive information presentation. According to Bürkle, this rather new functionality is, for example, available in the MED and the GDDS (<u>G</u>ießener <u>D</u>ata <u>D</u>ictionary <u>S</u>erver, [Ruan, Bürkle, et al., 2000]).

[30] The Medical Entities Dictionary (MED) is a medical data dictionary developed since 1988 by Cimino and colleagues at the Department of Medical Informatics of Columbia University (USA) [Cimino, James, 2000]. The source of the following example is the MED browser which is available online at: http://www.cpmc.columbia.edu/resources/med/browsers/wilcoxa/cgi-bin/mbrowser.cgi

Context in the sense of the GDDS is “the clinical information that is being presented at the moment the information need is occurring” ([Ruan, Bürkle, et al., 2000], p. 719). Context-sensitive information presentation in the GDDS allows relevant information sources (e.g. clinical guidelines) to be found for a given search term and is invoked by a so-called infobutton embedded in clinical applications. A generic algorithm determines the information sources that are to be displayed by exploiting the structure of concept representation. The GDDS is structured similarly to the UMLS in that concepts are grouped into concept classes (equivalent to UMLS semantic types). As in the UMLS, there exist relations between concept classes, which may hold between certain concepts of these classes. As an example, concepts of the class `drug` can have a `contains` relation with concepts of the class `drug substance`. Further, there is a concept class `information source`, the concepts of which are, for example, specific clinical guidelines.[31] Accordingly, any concept can be linked to relevant information sources. The algorithm for context-sensitive information presentation starts at a given concept (e.g. a concept `C` of the class `drug`) and returns all information sources which are linked to the given concept or to any of its directly or indirectly related concepts (in the example, this includes information sources attached to `C` itself as well as information sources linked to concepts of the class `drug substance` which stand in the `contains` relationship to `C`).

4.3 Situating the DD in the State of the Art

In the previous sections, we have reviewed existing terminologies and medical data dictionaries, which has shown the diversity in the structure of these systems. In particular, the way in which concepts are defined varies significantly.

Some systems, like the ICD, represent concepts (or rather terms, leaving the concepts implicit) as nodes in a hierarchy, in which the level of detail varies and the meaning of the hierarchical relation is not clearly defined. The meanings of the terms thus remain implicit to a great extent, posing problems with regard to the reproducibility of coding if different interpretations of the term are possible. Furthermore, although relations have become a common feature in terminologies, they are frequently mere links between concepts, leaving their meanings unspecified. As mentioned in section 4.1.4, for example, the UMLS provides an attribute for indicating the meaning of inter-concept relations, but only a small percentage of relations specifies a value for this attribute.

Other systems, like GALEN and SNOMED RT, achieve considerable improvements towards more explicit and precise definitions by employing formal languages for defining concepts. This includes explicit attributes and relations, as well as complex concepts which can be defined on the basis of elementary ones.

Considering the degree of precision of definitions required in the DD, it is obvious that the capabilities of terminologies like the ICD are insufficient. However, formal definitions as provided by GALEN and SNOMED RT do not seem to be fully appropriate for the purpose of the DD either. Specifically, having to represent a definition in a complex formal language may distract domain experts from the necessary initial clarification of a concept’s precise meaning since they cannot be expected to be familiar with the formal language. Therefore, it is reasonable to keep the definitions in the DD at a natural language level (as stated in the

[31] Note that this is an example of the problematic delimitation of objects and concepts as explained in section 2.4. In a strict sense, a concrete clinical guideline is actually an object.

requirement R1: Definitions) while at the same time allowing for formal components, e.g. explicit relations between concepts (cf. R11: Semiformal Content). This allows domain experts to discuss any differences in meaning without restricting them to the expressive limitations of a certain language.

Another issue is that existing systems are often designed for specific purposes. This can manifest itself as inflexible structure of the terminology (e.g. ICD and LOINC, which exhibit purpose-specific hierarchies and levels of granularity) or in the definitions of concepts themselves (e.g. medical data dictionaries contain the local vocabulary of an institution which is therefore adapted to specific local and organizational particularities). Both issues limit the possibilities for reusing terminologies in different applications and for different purposes than those intended during the development of the terminologies. Here, GALEN is the most advanced terminology and comes closest to the aim of describing concepts in a purpose-independent way.

Moreover, the overview of existing systems shows that none of them allows for the acquisition and representation of alternative definitions. Instead, all systems are concerned with providing a standardized terminology. However, as argued in the introduction and the requirements analysis, achieving consensus on the definition of a concept is not always desirable and may even be impossible (cf. the reasons in 3.2.1). The only system which comes rather close to providing alternative definitions is the UMLS since it integrates various terminology resources. However, as argued in section 4.1.4, the DD strives for greater precision in definitions than is available in the UMLS.

With the requirements analysis and a review of existing terminology systems at hand, the envisioned DD can now be related to the types of terminology systems defined in section 2.2. Since the DD is concerned with defining domain-relevant terms, it is a terminology. Further, it is additionally both a thesaurus since it will contain synonyms (cf. R3: Synonyms) and a glossary since it will contain definitions of terms. The DD will also integrate external nomenclatures and classifications. Although the DD is not primarily envisioned as a coding system, it can of course provide codes such that external applications can reference the concepts and alternative definitions contained in the DD.

Concerning the notion of medical data dictionary (MDD), it can be stated at this point that the application areas identified by Bürkle (see section 4.2) are in general also conceivable for the DD[32], although initially the emphasis lies on collecting and developing definitions of terms. It should further be noted that the borderline between terminologies and MDDs is not as strict as it might seem from the definitions given in section 2.2. Firstly, MDDs contain terminologies. Secondly, their delineation becomes less clear as the trend of development moves towards data dictionary servers which are independent from a particular hospital information system (referred to by Bürkle as the third generation of MDDs, cf. 4.2).[33] Against this background it can be concluded that the envisioned DD comes closest to what is referred to by Bürkle as a data dictionary of the third generation, since it will be independent of any particular hospital information system.

[32] apart from the last one which is text analysis, text production and translation of texts

[33] In fact, Bürkle classifies GALEN as an MDD of the third generation. We have nevertheless presented it in the section on terminologies since GALEN is not concerned with institution-specific terminology but rather aims at providing a widely reusable medical terminology and explicitly refrains from including any institution-specific issues which do not reflect consensus knowledge (cf. [OpenGALEN, 2004a]).

5 The Data Dictionary Model

In this chapter the specification of the DD model is presented. Section 5.1 describes the proposed solution by providing an overview of fundamental model decisions. After section 5.2, which contains some preliminary remarks on UML modeling and the definition of auxiliary classes necessary for the DD model, the subsequent sections provide a detailed specification of the DD model.

5.1 Proposed Solution

The most fundamental choice in the DD model is to adopt and extend the concept-centered approach. As explained in section 2.1, this approach distinguishes between terms and concepts, terms being language-dependent and concepts language-independent elements. We have decided to pursue this approach since, according to [Cimino, 1998], it is essential for dealing with multilinguality and synonyms.

Furthermore, because the DD needs to handle different alternative definitions of concepts, failure to represent concepts explicitly and instead defining terms directly (i.e. one term with multiple alternative definitions) would introduce another problem caused by ambiguity and synonymy. Defining terms directly would not allow for a distinction between alternative definitions of an ambiguous term with regard to the same meaning in contrast to definitions referring to different meanings. Thus, it would be impossible to determine automatically whether the alternative definitions should be compared in the harmonization process, which is obviously only reasonable for alternative definitions referring to the same meaning. Analogously, definitions describing the same meaning by using a synonymous term might be missed in the comparison of alternative definitions.

Note that our notion of concept in combination with alternative definitions differs slightly from that in the semiotic triangle. From a strictly extensional point of view, alternative definitions would actually always be understood as describing different concepts, since at least one object is classified differently due to the differences in definition. In view of the desired comparison and harmonization of definitions it is crucial, though, to group definitions which should be compared. A concept in the DD model is thus a means for representing the principal meaning that is common to alternative definitions which are to be compared during harmonization.

Another main idea is related to properly representing the kinds of differences in definitions. In section 3.2.1 the most frequent reasons which may cause differences in definitions were identified (they are listed again in table 5.1 below). We claim that these reasons can be further classified according to the kind of differences resulting from them. Above all, it is necessary to distinguish between

- differences in content (inconsistent alternative definitions),
- linguistic differences (consistent definitions, but different formulation or translation to different languages), and
- differences which are due to different points of view (consistent definitions, but different characteristics of a concept are described, possibly at different levels of detail).

Table 5.1 classifies the reasons discussed in section 3.2.1 according to these three kinds of differences (content, language, point of view).

Reasons for the occurrence of differences	Kind of difference
Reason 1: Disagreement due to Lacking Evidence	content
Reason 2: Different Conceptualizations according to Different Purposes (Different Classification, Different Is-A Granularity, Different Part-Of Granularity)	point of view
Reason 3: Interdisciplinary Character of Complex Domains	point of view, language[34]
Reason 4: Different Scientific Requirements	content
Reason 5: Different Specificity of Domain Context	content[35]
Reason 6: Organizational Differences	content
Reason 7: Different Geographical Location	content
Reason 8: Missing Standards	content
Reason 9: Linguistic Differences	language

Table 5.1: The reasons for the occurrence of differences (cf. section 3.2.1) can be classified according to three kinds of differences: content, language, and point of view.

We propose that not all of these kinds of differences should be represented equally by a separate alternative definition. Instead, alternative definitions should ideally be used to represent inconsistent definitions, while the consistent base definition which underlies a set of definitions that differ due to different points of view should be represented explicitly. Likewise, two definitions which are translations of the same content should be represented as such (cf. the requirement R5 on multilinguality).

The rationale for this is to avoid confusing substantially different definitions with definitions that are based on the same content but differ due to different purposes. More precisely, while one alternative definition (in the sense of a consistent basis) may have some scope of validity, e.g. an organization within which using the definition is mandatory, it may be permissible to use different variants of this definition according to purpose (i.e. different points of view, or a different translation).

[34] Reason 3 is classified as both, point of view and language, because the interdisciplinary character of complex domains leads to different points of view insofar as experts from various subject fields are interested in different information. Different formulation of definitions (language) may also be necessary in order to provide for different degrees of difficulty.

[35] Note that reason 5 “Different Specificity of Domain Context” is actually a special case, since a domain-specific definition may be based on a more generic definition, i.e. it is consistent with the generic definition but refines it in some aspects in order to adapt it to the more specific domain. This is however not specifically addressed in this thesis.

In summary, in contrast to the two levels available in the first version of the DD (cf. section 1.4), namely terms and alternative definitions, the DD model developed in this thesis proposes four levels:

1. terms,
2. concepts,
3. alternative definitions,
4. consistent variants of the same alternative definition (different points of view and translations).

5.2 Preliminaries

The DD model is specified using UML (Unified Modeling Language [OMG, 2003a]). Familiarity with UML is assumed since it is the standard modeling language for software systems and an introduction of all UML elements used in this thesis would exceed space limitations. For an introduction to UML the reader is referred to [Booch, Jacobson, et al., 1999].

Since this thesis is mainly concerned with the structure of the DD model, the model is described from a static point of view using UML class diagrams.[36] Operations are not specified since – due to the nature of the model – these are mostly operations for creating, adding and deleting the respective elements, which do not contribute to the solution of adequately representing the contents identified in the requirements. Further, it is an analysis model and will thus need further refinement for implementation (e.g. additional attributes and even additional classes may be necessary and, on the other hand, some classes could be merged for efficiency reasons, cf. [Balzert, Heide, 1999]).

Each of the subsequent sections focuses on describing one coherent part of the model and includes a UML diagram showing the relevant part of the UML model. For greater clarity, different colors are used in the diagrams (see figure 5.1). Brown color is used for classes, and those classes which are the focus of explanation in a diagram are shown in a more intensive shade than classes which are explained in detail in other diagrams but are included in the diagram because they are associated and necessary for comprehending a diagram. Blue color is used for objects.

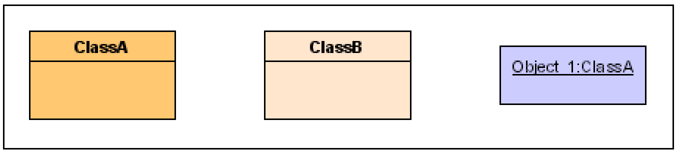


Figure 5.1: Shades of color used in the UML diagrams.

Regarding the font used in the explaining text, fixed-width font is used for names of classes, attributes, and associations when they are first introduced (e.g. `Term`, `Concept`, `Category`). However, for the sake of readability, after having introduced a class, its name will not be highlighted in special font unless ambiguities could arise.

[36] As an exception, one activity diagram is given in section 5.4.2 which illustrates the workflow for creating definitions.

When describing constraints on the model classes, the dot-notation according to OCL (Object Constraint Language [OMG, 2003b]) for specifying navigation to associated classes is used. As an example, if two classes A and B are related by an association r, the expression A.B denotes the set of instances of the class B which are related to an instance of class A. Alternatively, if class A has more than one association to B, rolenames are necessary at the association ends in order to indicate which of the associations is used in navigating to B. In this case, the respective rolename for the association end at B is used instead of the class name in the dot-notation. Likewise, rolenames are needed if r is a reflexive association in order to indicate in which direction the reflexive association is navigated.

Note that we define rolenames for association ends only if it is necessary with respect to unambiguity (i.e. for reflexive associations as well as with regard to the abovementioned dot-notation) and omit them otherwise due to space limitations. Instead, to improve comprehensibility, most association names are directed which is indicated by an arrow next to the name. These directed association names are not to be confused with navigability of associations (which would be indicated by an arrow-headed association). Since the DD model specified is an analysis model it is not concerned with implementational issues such as navigability, i.e. a directed association name does not imply that the association is navigable in that direction only.

Finally, before defining the DD model classes, three auxiliary data types are defined which are employed in the DD model (see figure 5.2). The values of the type LanguageT represent the languages in which the DD content (i.e. terms and definitions) as well as the graphical user interface (cf. requirement R27) is available[37]. Based on LanguageT, the standard data type string is extended to LexicalStringT which adds a language label to each string. Further, the type MultilingualStringT is a type whose values consist of multiple values of LexicalStringT (at most one per LanguageT) such that these strings are translations of each other. Values of MultilingualStringT are used in the DD model for storing translations of certain contents (cf. section 5.4.2).

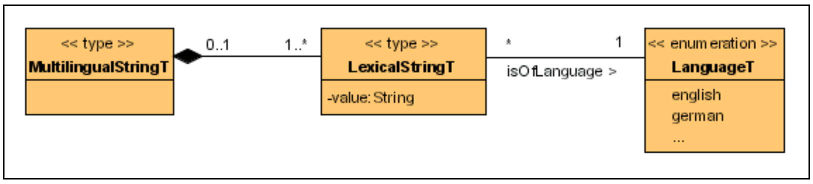


Figure 5.2: Data types defined for the DD model.

[37] If the set of languages in which the graphical user interface is implemented differs from the set of languages in which the content is developed, two subtypes of LanguageT will be necessary representing both sets. For the sake of simplicity, this is neglected here since it would require the definition of analogous subtypes for the other two classes as well.

5.3 Interrelations between Terms, Concepts, and Alternative Definitions

5.3.1 Concepts and Alternative Definitions

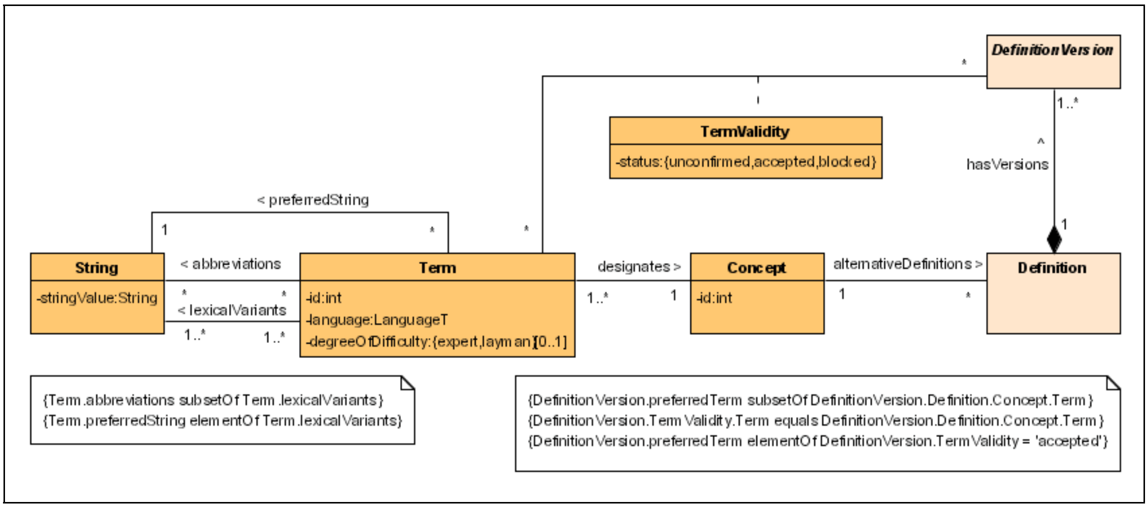

DefinitionVersion
*
1..*
TermValidity
-status:{unconfirmed,accepted,blocked}
^
hasVersions
< preferredString
1
*
*
1
String
-stringValue:String
< abbreviations
*
*
< lexicalVariants
1..*
1..*
Term
-id:int
-language:LanguageT
-degreeOfDifficulty:{expert,layman}[0..1]
designates >
1..*
1
Concept
-id:int
alternativeDefinitions >
1
*
Definition
{Term.abbreviations subsetOf Term.lexicalVariants}
{Term.preferredString elementOf Term.lexicalVariants}
{DefinitionVersion.preferredTerm subsetOf DefinitionVersion.Definition.Concept.Term}
{DefinitionVersion.TermValidity.Term equals DefinitionVersion.Definition.Concept.Term}
{DefinitionVersion.preferredTerm elementOf DefinitionVersion.TermValidity = 'accepted'}

Figure 5.3: The central classes of the DD model.

The diagram in figure 5.3 above shows the central classes of the DD model. We start with explaining the classes `Concept` and `Definition` because the definition of `Concept` is needed to understand the definition of the classes `String` and `Term`.

The fundamental requirement of the DD is to record existing alternative definitions of domain-relevant terms[38]. As described in the section on the semiotic triangle (section 2.1), terms are part of language and can have more than one meaning due to the occurrence of homonymy and polysemy. Thus, alternative definitions of the same linguistic term may refer to different meanings of the term. With regard to the analysis and harmonization of alternative definitions it is obvious that only definitions referring to the same meaning can be compared and harmonized. Therefore, we need to group alternative definitions according to the meanings they refer to. This purpose is served by the class `Concept`.

The class `Concept` can further be understood as an anchor node for `Definition`s that are defining the same meaning of a term – the definitions which are linked to the same `Concept` are `alternativeDefinititions` of that `Concept`. Each `Concept` can be defined by zero or more `Definition`s, and each `Definition` describes exactly one `Concept`. As described in the requirement R10 “Versioning”, the DD must support evolution of definitions. Therefore, each change to a definition results in a new `DefinitionVersion` (cf. section 5.4.2).

The next question is how to describe concepts, i.e. which attributes should be defined for the class `Concept`. In the concept-centered approach, concepts have exactly one definition which can thus be an attribute that describes the concept. This is unworkable in the DD, since multiple alternative definitions can exist for one concept. We therefore need to split the usual model entity of concept into the two classes `Concept` and `Definition` as described above. However,

[38] Note that in this section, ‘term’ is used in the sense as defined in the semiotic triangle. This is slightly different from the definition of the class `Term` in the diagram, which is explained in the next section.

some content that is usually part of a concept definition should not vary among alternative definitions of the same meaning, because it identifies the “principal meaning” (i.e. the part that all experts agree on, e.g. uncontroversial relations between concepts). Consequently, it must be decided which of those attributes that would usually describe a concept identify in our approach a “principal meaning” and should be shared by all alternative definitions (such attributes will be attached to the class `Concept`) and which elements may vary between the alternative definitions of one meaning (and are thus attached to the class `Definition` respectively). Examples of such decisions are explained in the following section on the representation of terms as well as in section 5.5.2 regarding relations.

A concept is identified by a unique identifier (attribute `id`). The set of terms that can be used to designate the concept serves as an indicator of the meaning. Note that the identifier must not carry any semantics because this is known to cause problems (e.g. many terminologies used to include hierarchical information in a concept’s identifier which is problematic if the hierarchy changes, but an identifier must remain stable). The `Concept` contains additional attributes and relations that should not vary among alternative definitions, e.g. the assignment to certain categories as described in section 5.4.2.

5.3.2 Details of the Representation of Terms

Regarding the representation of terms one could define a class ‘term’ based on the semiotic triangle that contained as instances all domain-relevant terms (i.e. linguistic labels which may consist of one or more words) the meaning of which is to be defined and discussed in the DD. The relationship between terms and concepts would then be represented by a many-to-many association in order to model that one term can have more than one meaning (i.e. ambiguity) and that the same meaning can be designated by several terms (i.e. synonymy).

This rather naive approach is problematic for a number of reasons. Firstly, lexical variants of terms (e.g. singular and plural) would be different terms and thus be treated as synonyms. Further, the choice of a preferred term would need to select one of the lexical variants, whereas it seems more reasonable to choose the preferred term independently of its singular and plural variants.

We therefore decide to adopt an approach that is analogous to the UMLS (cf. [NLM, 2003], p. 14 ff.). As shown in figure 5.3, the class `String` is defined which contains as instances all designations (i.e. a sequence of characters which form one or more words) the meaning of which is to be defined in the DD. The only attribute of `String` is `stringValue`, containing a case-sensitive sequence of characters. Examples are `Remission`, `remission`, `standard operating procedure`, `SOP`, `Standardarbeitsanweisung`, `Standardarbeitsanweisungen`.

The class `Term` groups all `String`s that are lexical variants of each other, e.g. `standard operating procedure` and `standard operating procedures`. A `Term` is also identified by a unique identifier (the attribute `id`) and contains references to all its lexical variants via the relationship `lexicalVariants`. The attribute `language` indicates the language of the term. The association `preferredString` allows for the selection of one `String` out of the `Term`’s `lexicalVariants` which is to be used as the default when displaying the `Term` (for example, the singular nominative long form of nouns).

The association abbreviations is a subset of the association lexicalVariants. It marks one or more Strings as being abbreviations of a Term. One Term can have multiple abbreviations because there may exist several possibilities of how to abbreviate a Term, e.g. leukocyte is sometimes abbreviated as leuko or as leu.

Note that from this definition it follows that Term is not concept-independent anymore: The association designates between Term and Concept is not a many-to-many association (as in the naive approach mentioned above) but a many-to-one association, i.e. one Concept can be designated by multiple Terms but one Term is linked to exactly one Concept. An example illustrating both variants is given in figure 5.4 below. The concept-dependency of Term necessarily follows from the need to distinguish Strings that are lexical variants of the same Term (e.g. plural and singular form) from Strings that are different Terms (e.g. white blood cell and leukocyte). The reason is that an ambiguous String (e.g. drug as a noun or as a verb) could have different lexical variants according to its different meanings (drug as a noun has a distinct plural form but not the verb). Thus, since the Term groups the lexical variants it must relate to exactly one meaning (i.e. concept). The example of SOP in figure 5.4 also illustrates that a String which occurs in multiple languages is a different Term, since it can also have different lexicalVariants in each language.

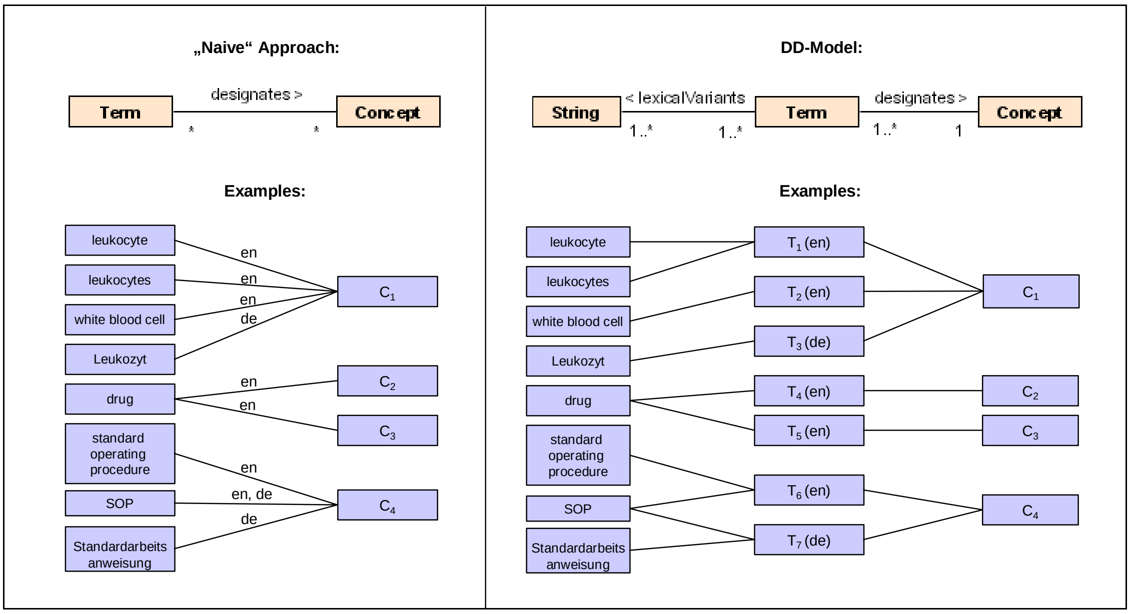


Figure 5.4: Comparison of the representation of terms in the naive approach and in the DD model. (The labels “de” and “en” are indicating the German and English language.)

A difference of our approach in comparison to that of UMLS is that we do not create artificial strings in order to disambiguate strings. For example, in the UMLS, for the ambiguous string drug, the additional strings drug <1> and drug <2> would be created in order “to give each meaning a unique name” ([NLM, 2003], p. 16). We do not see any reason why the creation of such artificial unique strings is necessary. Different meanings can always be distinguished by their unique id which can be concatenated to the ambiguous string.

In section 5.3.1 it was already mentioned that the classes `Concept` and `Definition` are “splitting” the usual concept of the common concept-centered approach. This was explained as being necessary in order to represent multiple alternative definitions of the same principal meaning. We have further mentioned that as a result of this splitting it must be decided for all information usually stored with the concept whether it may vary among different alternative definitions or not. This decision determines at which of the two classes this information is represented.

The attachment of terms to `Concept` instead of `Definition` is an example of such a decision. The set of terms which can be used to designate a concept is modeled as being independent of a particular alternative definition. We could have modeled terms as being dependent on a definition because different alternative definitions could also disagree on which terms are allowed to designate a concept. However, this would imply that each alternative definition needs to specify the complete set of allowed terms separately. We assume that this would result in high redundancy because most definitions will list the same synonyms, while the actual discord concerns rather the choice of a preferred term. For this reason we have decided to link the allowed terms to the class `Concept`. In the (presumably rare) case that a definition really does not agree with a term as an allowed designation, it may forbid the usage of certain terms which are in the set of allowed designations of the concept[39] by setting the value of `TermValidity.status` to `blocked`. In detail, each term that is newly assigned to a concept is valid “with reservation” for each alternative definition (i.e. `TermValidity.status` has the value `unconfirmed`). The editor(s) of each alternative definition may then choose to accept or not to accept this term:

- If not decided yet, `TermValidity.status` has the value `unconfirmed`. The editor must be prompted to decide on accepting the term.
- If not accepted, `TermValidity.status` is assigned the value `blocked`.
- If accepted, `TermValidity.status` is assigned the value `accepted`.

In contrast to this, the association `preferredTerm` is modeled as being dependent on `Definition`, i.e. each alternative definition can choose its own `preferredTerm` out of the set of `Term`s independently of other alternative definitions. This is desirable because we expect more disagreement on this choice than on the assignment of allowed terms described above. Note that since the terms are of multiple languages, there can be multiple `preferredTerm`s per definition, i.e. exactly one `preferredTerm` per language and per `degreeOfDifficulty` can be chosen per definition. (The attribute `degreeOfDifficulty` is introduced in section 5.7.2. In short, it indicates whether a term is an expert or layman term.)

Further details regarding the representation of terms are given in section 5.5.3 (lexical relations, i.e. synonymy and ambiguity) as well as in section 5.7.2 (concerning the attribute `degreeOfDifficulty`).

[39] Note that – as illustrated in the UML diagram – the association class `TermValidity` is actually attached to `DefinitionVersion` instead of `Definition` because it can change across versions. However, to keep the description understandable, we refer to the class `Definition` since the emphasis is on explaining which information is attached to `Concept` and which to `Definition`. The same holds for the relations `preferredTerm` and `preferredAbbreviation` which are defined below.

5.4 Alternative Definitions

In the following explanation of the modeling of alternative definitions, we recapitulate the main issues and requirements introduced thus far regarding this central model aspect, and relate these to the respective model elements.

5.4.1 The DD as a Repository of Definitions

As mentioned in section 1.2 of the introduction and further detailed in the requirements section (especially in R12), the idea of the DD is to establish a central repository which – similarly to a lexicon – provides term definitions. Unlike a normal lexicon (or more generally, a terminology), however, it must not only contain one definition per concept but the multiple different alternative definitions which are in use. As already explained in section 5.3.1, the multiple alternative definitions that exist for one concept are captured in the DD model by the class `Definition`. This fulfills the requirement of establishing a repository which contains the different definitions in use. In particular, this repository of definitions serves two purposes:

- Firstly, the collection of alternative definitions provides the basis for the harmonization process. The existing definitions can be compared and analyzed to determine the differences between them. Based on this analysis, consensus definitions can be developed.
- Secondly, even without having achieved consensus on definitions yet, the DD can already serve as a glossary because the alternative definitions collected from existing sources are actually valid and used by some organization.

In order to fulfill the second purpose of providing a glossary of valid definitions, two aspects are required in the DD model. Firstly, it must be possible to differentiate valid definitions from those which are still under discussion, e.g. the harmonized definitions being developed. This is accomplished by the versioning of definitions (represented by `DefinitionVersion`) which is controlled by a quality assurance cycle (cf. R16). Elaborating the quality assurance cycle is not part of this thesis but this issue is addressed by a parallel project. In short, different development status of a `DefinitionVersion` are differentiated. For example, the status ‘productive’ indicates a definition version which has passed the consensus process and can be used in documents. Likewise, a status for draft versions is needed, as well as for definitions which are not valid anymore (e.g. because they have been superseded by a consensus definition; note that content is never deleted but only marked as obsolete, as required in R10 about versioning).

Secondly, it is necessary to indicate the scope of validity of each definition in order to allow a user who is a member of a certain organization to retrieve only those alternative definitions which are valid within his or her organizational context. This is accomplished by the attribute `orgValidityComment` and the association `organizationalValidity` (see figure 5.5). The association `organizationalValidity` points to a set of organizations which are represented in the DD as an instance of the class `Community`.[40] However, since not all organizations will be represented in the DD as a `Community`, the attribute `orgValidityComment` can be used to

[40] For further details on the class `Community` see section 5.6. In short, an organization is only represented as a `Community` if it is developing content in the DD. Communities can be organized in an inclusion hierarchy which allows for the “inherited” validity of a definition in the included communities.

enter a textual description indicating the organizational validity. Although this cannot be interpreted automatically, it is useful information for human users. Both items of information (the attribute and the association) are optional and can also be used in combination, e.g. the `organizationalValidity` `KML`[41] indicates the broad organizational scope, and `orgValidityComment` provides the detailed information on the specific study group (which may not be represented as a community) for which a definition is valid.

Furthermore, as described in reason 7 in section 3.2.1, definitions are sometimes influenced by country-specific particularities, e.g. different laws. In order to support the correct reuse of definitions, it should be possible to indicate if a definition is reusable within some specific geographical area only. Therefore, the association `geographicalValidity` is defined which allows for restricting the validity of an alternative definition to a set of `GeographicalArea`s. A `GeographicalArea` can be, for example, a continent, a country, etc. They can be arranged in an inclusion hierarchy, from which always the broadest area in which a definition is valid should be chosen, meaning that the definition is also valid in all included geographical areas. Note that although such an inclusion hierarchy can also be defined for communities, the “inheritance” of validity is handled differently in communities as described in section 5.6.

Although it may seem useful to model an association between communities and geographical areas which represents the location of a community and could – in addition to the organizational validity – be used for the selection of definitions valid for the organization, we have decided not to do so. The rationale is that, for example, although the organization KML is located in Germany, whether a definition with a validity restricted to Germany can be used within a clinical trial depends on the geographical scope of that trial. If it is an international trial, the definition which is valid only within Germany cannot be used. On the other hand, if the trial is conducted Germany-wide only, this definition may be used. Thus, it cannot always be assumed that if a community is located in some geographical area, it can always use definitions which are valid within this area.

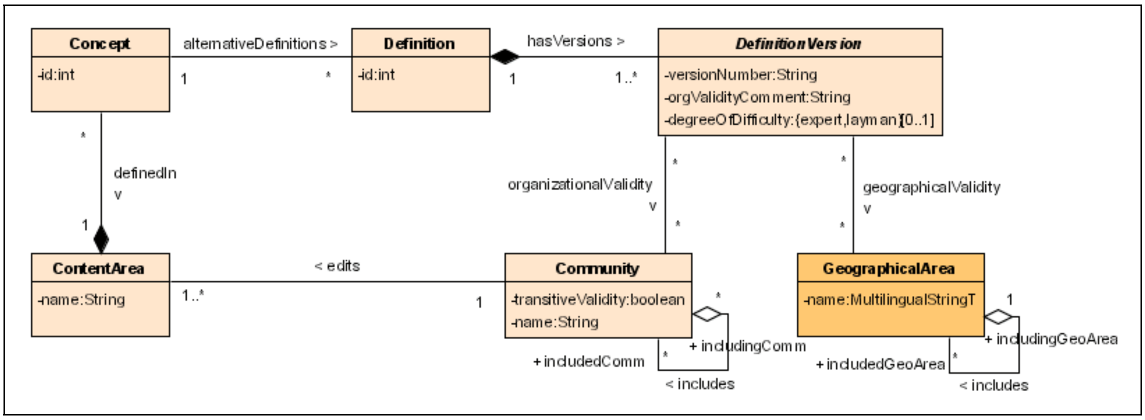


Figure 5.5: Validity of definitions.

[41] referring to the “Competence Network Malignant Lymphomas” (KML) as introduced in section 1.3

It is important to understand that the definitions in the DD will be under constant development, which is not only necessary because of changes due to harmonization but also due to progress in research. At each point in time the DD contains valid definitions as well as definitions under development which, as soon as they become valid, may replace other definitions which become `obsolete` as the result of harmonization.

Note further that a concept may have – apart from the definitions which are substituted by the harmonized definition – additional alternative definitions which remain valid alternative definitions after harmonization. Consider for example the case of a concept having four alternative definitions: one generic definition cited from the ICH Good Clinical Practice (GCP) [ICH, 1996] and three definitions from three cooperating study groups. The study groups aim at harmonization and therefore want to unify the three study group definitions. The definition from ICH GCP is also analyzed during the development of the harmonized definition (in order to create a harmonized definition which is maximally compliant with the generally valid definition from ICH GCP), but this definition of course cannot be changed substantively by the DD members. It remains valid even after finishing the harmonization process of the three definitions resulting in a new harmonized definition, and is therefore kept in the DD to fulfill the aim of maintaining a repository.

5.4.2 Structure of Definitions

Recall the main purposes of the DD explained in section 5.4.1. On the one hand, existing definitions are collected in the DD in order to analyze and compare them. This collection of definitions serves as a repository of valid definitions which can be queried by other applications. On the other hand, the DD supports the development of improved and ideally harmonized definitions.

Keeping these issues in mind, recall the requirements from chapter 3.1 regarding the format of definitions. According to R1, the DD should contain precise natural-language definitions which could be quotations from available documents. This requirement can be mapped to the first purpose of collecting existing definitions since this purpose requires a high degree of flexibility in the format of definitions (the format and precision of existing definitions varies considerably). On the contrary, it is mentioned in R2 that it is useful for definitions to be created using pre-defined templates in order to achieve uniformity among definitions of similar concepts. The use of templates supports the purpose of developing harmonized definitions since a uniform structure of definitions facilitates the comparative analysis of alternative definitions.

In order to satisfy both requirements, we argue that it is not desirable to enforce the use of templates for the entry of definitions: since quotations will most likely not fit into the template structure, it should be possible to enter these as free text. Thus, when entering a definition[42] the

[42] Note that "entering a definition" in this context actually means to enter a new `DefinitionVersion`, since, as described in the previous section on versioning, the actual content of a definition is stored in an instance of `DefinitionVersion` and each editorial change of a definition (i.e. a `DefinitionVersion`) results in a new `DefinitionVersion`. However, for the sake of simplicity, we will, if the context is clear, always refer to a "definition" instead of a `DefinitionVersion` even though the latter is actually more correct from the point of view of the model. In cases where we really intend to refer to the model class `Definition`, we always use the fixed width font, as in this sentence.

editor can choose between two formats: a free-text definition or a structured definition which is based on templates. Ideally, however, all newly developed definitions should use the structured format and only existing definitions which are quoted should use the free-text format.

In the class diagram in figure 5.6, the two format options are represented as subclasses of `DefinitionVersion`. The subclass `InformalDefVersion` represents the free-text format and the subclass `SemiformalDefVersion` represents the structured definition which is based on templates.

Note that since these classes are subclasses of `DefinitionVersion` instead of `Definition`, the format of an alternative definition can change between versions. This is reasonable since the first `DefinitionVersion` of a `Definition` could be a quotation and thus use the free-text format, while the second version tries to improve the accuracy of the definition by transforming the content of the definition into the structured format.

Multilinguality of Definitions

Before explaining the two possible structural formats of definitions, it is necessary to consider the issue of multilinguality of definitions. Requirement R5 asks for the availability of translations of definitions in different languages and for the possibility to distinguish two definitions which are translations of the same content from definitions which are different in content. In the DD model, translations of a definition to different languages are therefore not entered as different alternative definitions, but are all included in one `DefinitionVersion` of an alternative definition, i.e. each `DefinitionVersion` can be considered as having some language-independent content which is made available in several languages. As a consequence, all translations of a `DefinitionVersion` have the same format (i.e. either informal or semiformal) which is reasonable since this helps to ensure that they are exact translations of the same content.

Note that as mentioned in R5, translating a definition is not mandatory; it is possible that the first version of a definition is available in one language only and is translated to another language in the next `DefinitionVersion`. When editing one `DefinitionVersion` it is important to keep all translations of this version consistent, i.e. substantive changes should always be made for all existing translations, and together these changes result in a new version of the definition rather than effecting the change for each translation separately. Ensuring this should be supported by the user interface. How the translations of a `DefinitionVersion` are actually stored in the DD depends on the format of the definition and is described in the respective following subsections.

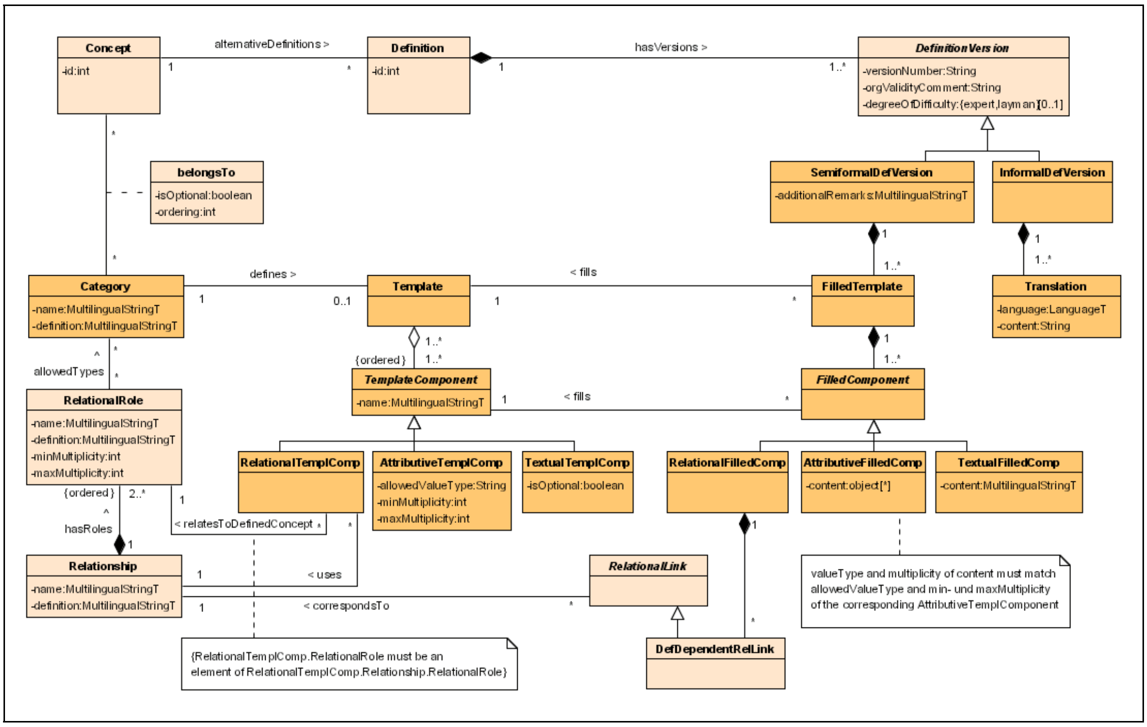


Figure 5.6: Structure of definitions: informal and semiformal definitions.

Details of `InformalDefVersion`

The representation of an informal definition is straightforward. An informal definition version is represented by the class `InformalDefVersion` and consists of one or more `Translations` each describing the same substantive definition in one language. A `Translation` is described by the attribute `content` containing the free-text definition and an attribute `language` specifying the language of this `Translation`. As a constraint, each `InformalDefVersion` may have at most one `Translation` per language.

Details of `SemiformalDefVersion`

The representation of a structured, i.e. semiformal, definition is more complex than the informal format described above.

The basis of a `SemiformalDefVersion` is the assignment of a `Concept` to one or more instances of `Category`. (The reason a concept is allowed to belong to multiple categories instead of only to one is explained in the section on contexts and views (5.7.1), in particular in the explanation of the view axis `V_Category`.) A `Category` represents a group of similar concepts (recall the examples given in R2: `laboratory data`, `processes`, `diseases`, `classifications`) and may have an associated `Template` which defines a uniform schema for `SemiformalDefVersions` that use (i.e. fill) this template.

Each `Template` consists of one or more `TemplateComponents`, where each `TemplateComponent` contains the definition of one section of the template. A `TemplateComponent` has an attribute `name` – the “heading” of this section – which will be displayed when the user is prompted to fill this template component. In order to fulfill the requirement of multilingual definitions, the name is of data type MultilingualStringT (introduced in section 5.2). The name of the section is thus displayed in the language of the definition which is being entered.

Template components can be reused in different Templates, thus this composition is a many-to-many relationship. Further, the set of template components that constitutes a template is ordered in order to allow for the representation of a logical order in which the definition components follow each other to make up the whole definition. The object diagram in figure 5.7 below illustrates the definition of categories and their templates; the template consisting of template components is depicted on the right and the corresponding objects are indicated by the dashed lines.

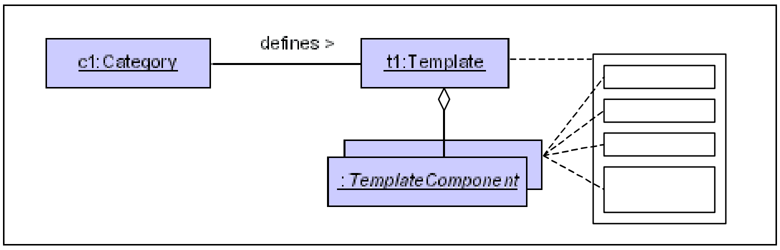


Figure 5.7: Illustration of the relation between a category and its template consisting of template components.

Furthermore, R2 demands the possibility to restrict the value type of the content which may fill a section. TemplateComponent thus has three subclasses with different properties regarding the allowed value type (cf. figure 5.6):

- A TextualTemplComp defines a section which will be filled with language-dependent text, i.e. that needs to be translated into different languages. The Boolean attribute isOptional indicates whether filling a given section is optional or mandatory.
- An AttributiveTemplComp defines a section which will be filled with language-independent values, i.e. values that do not need to be translated (e.g. numbers or strings such as measurement units: m, kg, etc.). It has the attribute allowedValueType which specifies the value type of the content of each AttributiveFilledComp which fills this template component; possible values of this attribute are the common data types, e.g. string, integer, float. The attributes minMultiplicity and maxMultiplicity specify how many values are allowed when filling a section (i.e. this is a dynamic multiplicity restriction for the attribute content of the class AttributiveFilledComp). Note that the multiplicity implicitly determines whether filling a section is optional, i.e. it is optional if minMultiplicity equals zero.
- A RelationalTemplComp defines a section which will be filled with links to concepts, i.e. relational links[43] are created between the concept being defined and the

[43] These relational links are instances of the class DefDependentRelLink, a subclass of RelationLink. For a description of relations see sections 5.5.1 and 5.5.2. The classes Relationship and RelationalRole are also defined in detail in those sections – however, the class names should be intuitive enough that their meanings are understandable with regard to RelationalTemplComp described in this section.

concepts entered into the section. The Relationship (e.g. is-a, part-of, symptom-of) which is used for creating the relational links is specified by the association uses. A RelationalTemplComp is similar to an AttributiveTemplComp insofar as it is language-independent and has a value type and multiplicity. However, the value type and multiplicity do not need to be defined explicitly since they can be derived from the definition of the Relationship used. Specifically, the value type is determined by the categories which are allowed to fill the arguments (RelationalRoles) of the chosen Relationship, i.e. only those concepts can fill a relational template component which belong to at least one of these categories. An example of the use of a relational template component is given in section 5.5.2, concerning concepts of the category laboratory parameter measured by certain measurement procedures.

While instances of Template and TemplateComponent define a schema for definitions, the classes FilledTemplate and FilledComponent capture the content which is entered into such a schema. In detail, each SemiformalDefVersion consists of a set of FilledTemplates, where each FilledTemplate corresponds to exactly one Template (via the association fills). Analogously, each FilledComponent corresponds to exactly one TemplateComponent and contains the content which conforms to the template structure implied by the corresponding TemplateComponent. Together, the content contained in the FilledComponents which are in turn contained in the FilledTemplates form a SemiformalDefVersion.

Having explained the rather complex representation of Templates and TemplateComponents, let us return to the beginning of this subsection where we stated that the basis of a SemiformalDefVersion is the assignment of a Concept to one ore more instances of Category. Thus far, we have not yet explained in detail the association class belongsTo which represents the assignment of a concept to a set of categories. In particular, we need to clarify how exactly a SemiformalDefVersion is constructed from a set of assigned categories, and which meaning the attribute isOptional of the association class belongsTo has in contrast to individual optional template components of a category template. The activity diagram in figure 5.8 below illustrates the steps for creating a new DefinitionVersion and what it means for the creation of a SemiformalDefVersion if a category assignment or individual template components are optional.

At the initial state at the top of the diagram, the workflow starts with the creation of a new instance of DefinitionVersion for a given Concept c which belongsTo a set of n categories (denoted by c.category). For each category there exists at most one template[44], consequently, the number of templates which are available for a semiformal definition (denoted by c.category.template) may be smaller than the number of categories to which the concept belongs. Creating a SemiformalDefVersion is only possible if at least one of the categories to which the concept belongs defines a template; otherwise, i.e. if no template exists,

[44] Note that it is optional for a Category to define a Template. A category without a template is just like a "label" to group similar concepts and does not influence the structure of a SemiformalDefVersion.

only an InformalDefVersion can be created. This choice is illustrated in the activity diagram by the first branch (the diamond symbol). If at least one template exists, the editor of the definition may choose which format to use (represented by the activity state choose format).

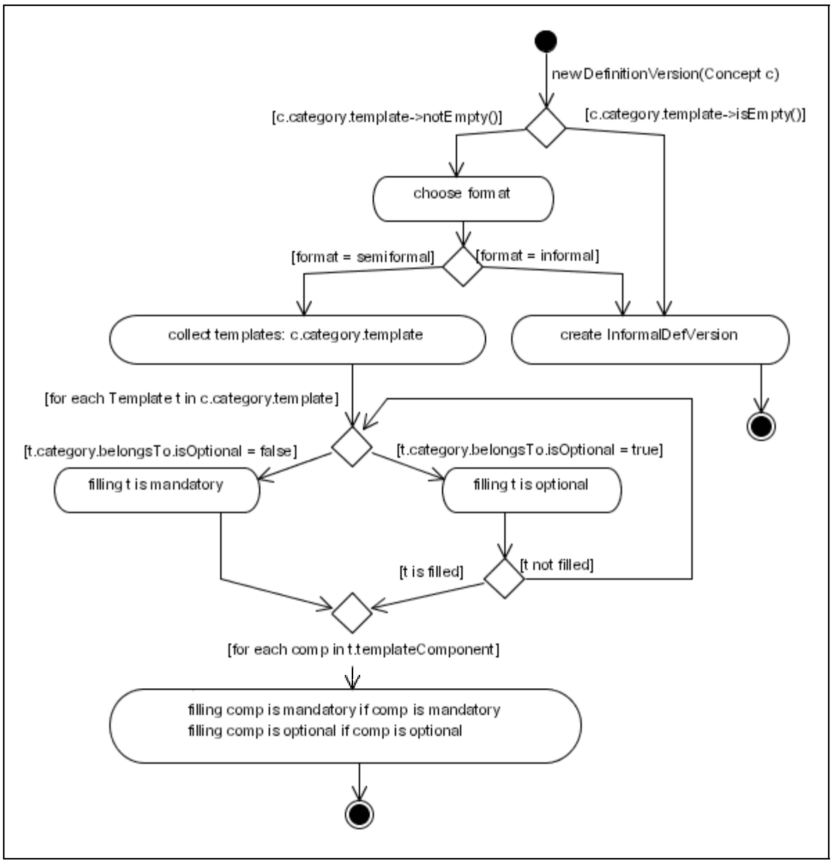

newDefinitionVersion(Concept c)
[c.category.template->notEmpty()]
[c.category.template->isEmpty()]
choose format
[format = semiformal]
[format = informal]
collect templates: c.category.template
create InformalDefVersion
[for each Template t in c.category.template]
[t.category.belongsTo.isOptional = false]
[t.category.belongsTo.isOptional = true]
filling t is mandatory
filling t is optional
[t is filled]
[t not filled]
[for each comp in t.templateComponent]
filling comp is mandatory if comp is mandatory
filling comp is optional if comp is optional

Figure 5.8: Activity diagram for creating a new instance of DefinitionVersion[45].

Only in case the semiformal format is chosen does the attribute isOptional of the association class belongsTo become relevant and determines whether a specific template must be filled (if isOptional=true) or not. It should be emphasized that this attribute does not hold "globally" for all DefinitionVersions but only for SemiformalDefVersions. The rationale is given at the beginning of this section where it is explained that it should also be possible to enter quotations of definitions which might not fit into any template structure. Similarly, a mandatory template component is not "globally" mandatory for the whole definition but only mandatory within the template in which it is contained, i.e. only if the template is filled (which may be optional or not as explained above) must all mandatory template components be filled. For the sake of flexibility, each semiformal definition contains a field for additional remarks (attribute additionalRemarks) which can be used for entering any information which is considered important but does not fit into the template(s).

[45] Although the last state in the activity diagram itself represents a complex activity, this is deliberately not represented in detail because the higher complexity would detract from the essential information to be conveyed by the diagram.

Figure 5.9 below exemplifies the filling of templates according to the steps described above. Mandatory templates and template components are indicated by a thick line. The template components which are filled by the semiformal definition are shown in gray; those which are not filled are shown in white. Note that of course only filled – i.e. gray – template components are instances of `FilledComponent` (indicated by the dashed lines), while no instances need to be created for those template components which are not filled since nothing needs to be stored in such cases.

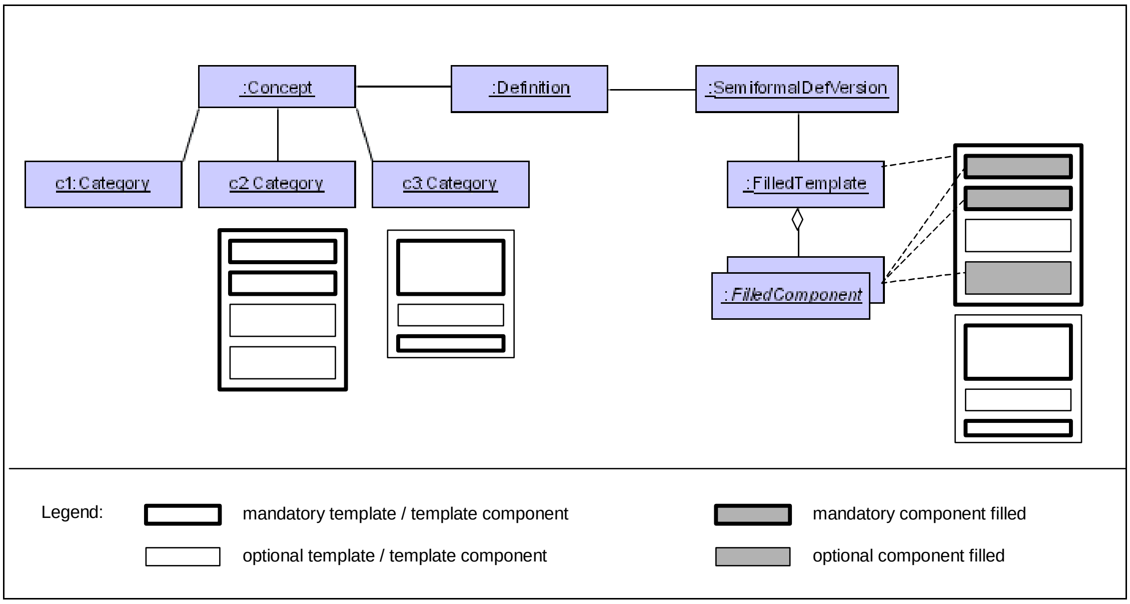

:Concept
:Definition
:SemiformalDefVersion
c1:Category
c2:Category
c3:Category
:FilledTemplate
:FilledComponent
Legend:
mandatory template / template component
optional template / template component
mandatory component filled
optional component filled

Figure 5.9: Illustrating the connection between templates and filled templates. The concept is assigned to three categories (`c1`, `c2`, `c3`), only two of which define templates, one of which being mandatory (i.e. that of `c2`). The semiformal definition must therefore fill at least the mandatory components of the mandatory template.

A remark concerning the translation of a `SemiformalDefVersion`: as mentioned in the description of the different subclasses of `TemplateComponent`, only textual template components are language-dependent while attributive and relational template components are language-independent and thus do not need to be translated. Hence, in order to translate a `SemiformalDefVersion`, it is only necessary to translate the `content` of all `TextualFilledComp`s by adding a string of the desired language to the `content` attribute which is stored as a `MultilingualString`, as well as to translate the `name` of the respective `TemplateComponent`s and the `additionalRemarks`. The rest of the `SemiformalDefVersion` does not need to be translated since it is language-independent. This solution provides maximum support for ensuring that the translations of a semiformal definition are equivalent in content, and minimizes the effort needed for manual translations.

Last but not least, the issue of storing bibliographical references for definitions as called for by R1 is addressed. Figure 5.10 below shows the relevant classes in the UML diagram. Each `SemiformalDefVersion` and each `Translation` of an `InformalDefVersion` has exactly one `BibliographicalAnnotation` which contains the bibliographical information for a definition. Two types of definitions are conceivable: direct quotations and definitions which are formulated by DD editors, possibly with references to existing sources. Accordingly, two subclasses of BibliographicalAnnotation reflect these types: Quotation and OwnDefinition. The class BibliographicalSource represents a structured data type whose values represent bibliographical sources, e.g. books, documents, etc. The structure of this data type is not further elaborated herein. The rationale is that the implementation may consider linking a reference management system to the DD, which will determine the detailed structure of BibliographicalSource.

A Quotation is citedFrom exactly one source. In order to allow for indirect quotations (i.e. a definition that is quoted from a source could be itself a quotation from another source), it is additionally possible to enter a second source (citedFrom) which represents the source from which the quotation originates. The more frequent use case which is expected here relates to connecting the DD to the UMLS (cf. section 4.1.4). Definitions which are integrated from the UMLS will be citedFrom the UMLS, but since the UMLS is itself assembled from definitions of its source vocabularies, the source vocabulary from which a definition originates can be stored via originalSource.

An OwnDefinition may be based on multiple bibliographical sources, which are stored via the association references. The authors of a definition can be entered as String values in the attribute authors or reference registered users of the DD via the authors association.

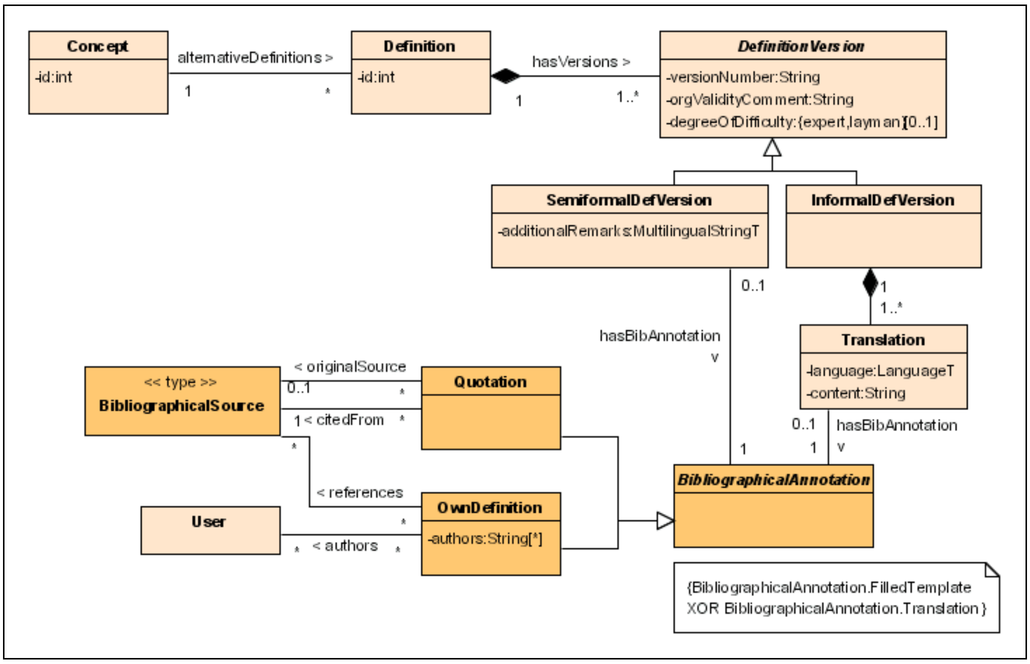


Figure 5.10: Bibliographical annotations of definitions.

5.5 Relations

Requirement R6 calls for the possibility to define relations. When looking at the examples of relations given in R6-Ex1, it becomes obvious, however, that the requirement does not distinguish between different kinds of relations. While the relations synonym-of and hypernym-of are relations that hold between terms and thus depend on language; part-of, is-a, causes, and measures hold between concepts and are valid independently of language.

Since the DD model distinguishes between terms and concepts, we consequently also distinguish between two different kinds of relations, which we call **lexical relations** and **semantic relations**, respectively. Although this distinction is obvious, it is not made by all terminology systems since many systems do not distinguish between terms and concepts.

Within this thesis, we concentrate on semantic relations because they contribute to the definition of a concept and are thus more important than lexical relations with regard to the aim of developing explicit and precise definitions. Regarding lexical relations we will only deal with those which are most important for a terminology, e.g. synonymy and ambiguity; additional lexical relations are left for further development. How semantic and lexical relations are represented is described in the following two sections.

5.5.1 Semantic Relations

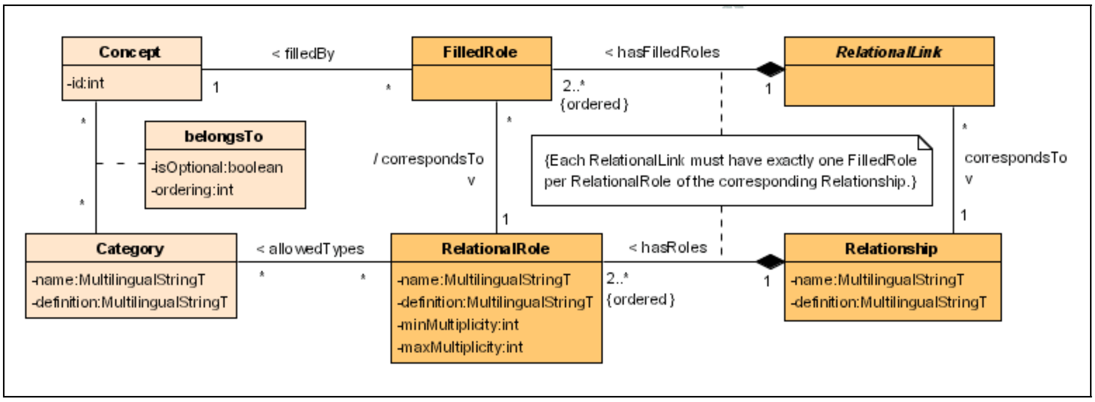

Concept
-id:int
< filledBy
1
*
FilledRole
< hasFilledRoles
2..*
{ordered}
1
RelationalLink
*
belongsTo
-isOptional:boolean
-ordering:int
*
/ correspondsTo
v
*
1
{Each RelationalLink must have exactly one FilledRole per RelationalRole of the corresponding Relationship.}
*
correspondsTo
v
1
Category
-name:MultilingualStringT
-definition:MultilingualStringT
< allowedTypes
*
*
RelationalRole
-name:MultilingualStringT
-definition:MultilingualStringT
-minMultiplicity:int
-maxMultiplicity:int
< hasRoles
2..*
{ordered}
1
Relationship
-name:MultilingualStringT
-definition:MultilingualStringT

Figure 5.11: Representation of semantic relations.

Figure 5.11 shows the part of the DD model which specifies the representation of semantic relations. As determined in requirement R6 it should be possible that relations can be defined first and selected later when establishing a relation between concrete concepts. This requirement is fulfilled by two basic classes shown in the right part of the UML diagram above: `Relationship` and `RelationalLink`. In short, the first captures the definition of a reusable relationship, and the latter represents the usage of a specific relationship to create a relation between specific concepts. In other words, the definition of relations allows for the creation of a semantic network of concepts, where the `RelationalLink`s are the edges between the `Concept`s as nodes and each `RelationalLink` `correspondsTo` one predefined `Relationship` which describes the type of the relational link. In the following, these and related model classes are described in detail.

A `Relationship` is described by the attribute `name` (which is again a `MultilingualString` so that it can be translated) and its `RelationalRole`s. A `RelationalRole` is like a placeholder which is filled at a concrete `RelationalLink` by a `Concept`. The class `FilledRole` represents such a filled `RelationalRole` by maintaining a reference to the `RelationalRole` filled (via the association `correspondsTo`) and a reference to the `Concept` which fills this role. Each `FilledRole` is of course part of exactly one `RelationalLink`. Duplicate `RelationalLink`s are not allowed, i.e. each `RelationalLink` which is essentially an n-tuple of concepts must be unique with respect to the corresponding `Relationship`.

Note that relationships are not restricted to binary relationships: our approach is capable of representing relationships of arbitrary arity because the roles of a relationship are defined as separate classes instead of as attributes of the class `Relationship`, and each relationship can have two or more `RelationalRole`s (the multiplicity is specified as `2..*`). Binary and ternary relationships (with two or three relational roles, respectively) are expected to occur most frequently.

The object diagram in figure 5.12 below shows example instances of the classes `Relationship` and `RelationalRole`.

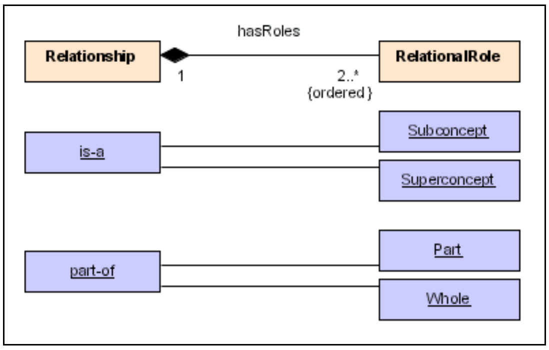


Figure 5.12: Examples of instances of `Relationship` and `RelationalRole`.

Further, the class `RelationalRole` has an association `allowedTypes` to `Category`. This association can be used for restricting the categories of concepts which may fill a `RelationalRole`. Restricting the category is optional, however, since for example the `is-a` relationship can be defined between all categories of concepts (i.e. no restrictions are needed). Note that the name of the `RelationalRole` is allowed to be the same as the name of the allowed `Category` but will usually be different, because multiple categories can be given as `allowedTypes` thus we need to specify a name for the role. Further, we need distinct role names in cases where the `allowedTypes` are the same for both roles. For example, in the case of `part-of` we could define a `Category Substance` which would be used as the `allowedType` for both `RelationalRole`s – hence we need different role names.

Since multiple categories can be specified as `allowedTypes` and one `Concept` may belong to multiple categories, we must define precisely what it means if more than one `Category` is given for `allowedTypes`. In order not to be too restrictive, we suggest that this case should be interpreted in the sense of an OR-restriction, i.e. the concept must belong to at least one of the categories which are specified as `allowedTypes` and may additionally belong to other categories which are not included in `allowedTypes`. If no categories are specified as `allowedTypes` for a `RelationalRole`, then all concepts – no matter which category they belong to – may fill this role. More formally:

> A `Concept C` may fill a `RelationalRole R` if and only if
> (`R.category` $= \emptyset$) or (`C.category` $\cap$ `R.category` $\neq \emptyset$).

The last characteristic of the class `RelationalRole` that needs to be described are the attributes `minMultiplicity` and `maxMultiplicity`. They define restrictions on instances of

`RelationalLinks` and are defined in analogy to the multiplicity of associations in UML. In UML, multiplicity in general is defined as a “specification of the range of allowable cardinality values – the size – that a set may assume” ([Rumbaugh, Jacobson, et al., 1999], p. 346). With regard to n-ary associations which can be compared to `Relationship`s in the DD model, multiplicity of an association end in UML (which maps to `RelationalRole`) is defined more precisely as “the potential number of values at the end, when the values at the other n-1 ends are fixed” ([Rumbaugh, Jacobson, et al., 1999], p. 350). What is meant by values in this definition are in the DD model the `Concept`s which take part in `FilledRole`s. We adopt the above definition without substantive change and only adapt it to the classes in the DD model:

> Given is a `Relationship` which has `n` `RelationalRoles`. The attributes `minMultiplicity` and `maxMultiplicity` define the following constraint on the set of instances of `RelationalLink` that correspond to the same `Relationship` (remember that each of these `RelationalLink`s links `n` concepts where each concept fills exactly one of the `n` `RelationalRoles`):
>
> The potential number `m` of `Concept`s that may fill the `RelationalRole r`, given that the fillers of the other `n-1` `RelationalRole`s are fixed, is constrained such that `[r.minMultiplicity` ≤ `m` ≤ `r.maxMultiplicity]`.
>
> Representing unrestricted `maxMultiplicity` should be possible as well; this could be implemented by a special integer value, e.g. `-1`.

Although this definition sounds complicated, it is widely used in UML diagrams and is therefore considered to be appropriate. The advantage of this definition is that it is valid for relations of arbitrary arity (cf. [Rumbaugh, Jacobson, et al., 1999], p. 350). An example will clarify the definition. Reconsider the example of the `is-a` relation, which can be defined as an instance of `Relationship` having the two `RelationalRole`s `Subconcept` and `Superconcept` as illustrated in figure 5.13 below. The multiplicity defined in this example at `Superconcept` means that, given that the other end is fixed, i.e. given a concrete `Subconcept`, at least zero and at most one concept may fill the `Superconcept` role for this given `Subconcept`; in other words each `Subconcept` has at most one `Superconcept`. The multiplicity defined at `Subconcept` means that each `Superconcept` may have an arbitrary number of `Subconcept`s, including zero. If the `is-a` relation should be defined such that it allows for an unrestricted number of `Superconcept`s, the `maxMultiplicity` at `Superconcept` would be set to `-1`.

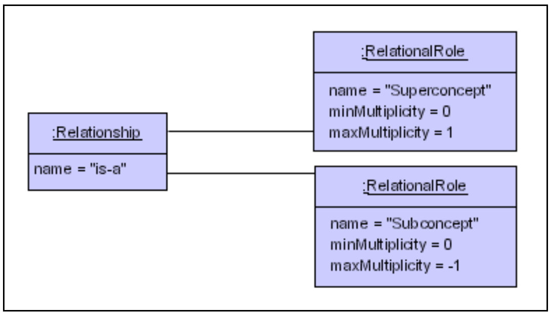


Figure 5.13: Example of the definition of multiplicities.

The modeling of relations is a complex topic and many additional features are conceivable, e.g. the differentiation of different kinds of relations (hierarchical, transitive, symmetric, etc.) or the representation of relation hierarchies. Dealing with each of these issues in detail would lead us too far astray and is thus left for further development.

The model developed thus far seeks to provide a compromise between expressiveness and usability, i.e. it attempts to cover those issues which are of greatest relevance to the DD while still keeping the complexity at a level adequate to be understood by domain experts. Although the idea of defining groups of concepts (in the DD model: categories, in UMLS: semantic types, in GDDS (cf. section 4.2): concept classes) and using them for defining relations is common to many terminologies, most approaches allow for defining binary relations only and do not support multiplicity constraints.

5.5.2 Semantic Relations and their Validity in Alternative Definitions

In the previous UML diagrams, `RelationalLink` is defined as an abstract class (the class name is in italic font). In this section, the two concrete subclasses are defined and explained.

As mentioned, semantic relations contribute to the definition of a concept – a concept is often defined in relation to other concepts (e.g. by referring to a broader meaning which is then constrained). As already elaborated in section 5.3.1, the classes `Concept` and `Definition` are "splitting" the usual concept of the common concept-centered approach and thus it must be decided for each kind of information usually stored with the concept whether it may vary among different alternative definitions. This was relevant in section 5.3.2 when we had to deal with the decision of whether terms should depend on alternative definitions or be attached to the class `Concept` and thus be independent of alternative definitions. Since we expected little disagreement on the set of terms that are allowed to designate a concept, we chose to attach the allowed terms directly to `Concept` instead to `Definition`. In order to still allow a `Definition` to be incompatible with a term, we introduced the class `TermValidity` which can be used by a definition to block a specific term.

Semantic relations pose the same modeling problem: should alternative definitions define the relations (which would allow disagreement on the question whether a relation is valid) or should relations be defined directly between concepts, independently of alternative definitions, and thus not allow for disagreement?

The desire to use relationships in semiformal definitions as described in 5.4.2 (relational template components) speaks in favor of the first variant. In this case, `RelationalLink`s are part of exactly one definition. However, there are other examples of relationships, e.g. the `is-a` relationship, whose validity is uncontroversial and thus it would result in a high degree of redundancy if this relational link needed to be defined by each alternative definition separately. Since both options are useful, we choose to provide both. Therefore, two subclasses of `RelationalLink` are defined, each of which corresponds to one model variant. See figure 5.14 for the UML diagram.

`DefIndependentRelLink` defines a relational link which is uncontroversially valid for all alternative definitions and is thus established directly between concepts independently of alternative definitions. In analogy to `TermValidity` (see section 5.3.2), however, the class `RelationValidity` is defined in order to provide the possibility to block a `DefIndependentRelLink` in case an alternative definition does not agree to its validity. On the contrary, a

DefDependentRelLink is valid only within the scope of a specific alternative definition. This is represented in the model such that a DefDependentRelLink is contained in a RelationalFilledComp which is part of the definition within the scope of which the relational link is valid.

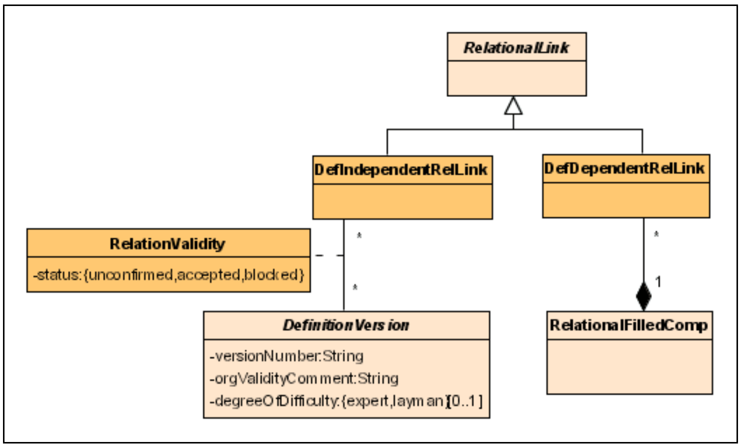


Figure 5.14: Types of relational links.

Before giving examples for both types of relational links, it needs to be explained how the multiplicity of a Relationship is to be interpreted for DefDependentRelLinks. As can be seen from the UML diagram, a DefDependentRelLink is part of a RelationalFilledComp of a definition. When filling this RelationalFilledComp, the full multiplicity can be used, i.e. each alternative definition can – independently of any other definition – exhaust the full multiplicity. The other option would be to understand the multiplicity for all alternative definitions of a concept at once so that all relational links of a relationship, which are defined at any alternative definition of the concept, are counted together and must obey the multiplicity. This would not make sense, however, as will become clear from the following examples.

As an example of a DefIndependentRelLink, consider the concept <clinical trial> and the more specific concept <phase-I-trial>. A RelationalLink of the Relationship is-a can be defined between these concepts, and since this fact is uncontroversial, the type of this relational link could be DefIndependentRelLink.

As an example of a DefDependentRelLink, consider laboratory parameters which are measured by measurement procedures. For most laboratory parameters there exist alternative measurement procedures which differ in various aspects, and different laboratories often use different procedures. However, with regard to harmonization, it is desirable to use the same measurement procedure in all centers of a multi-center trial. Laboratory parameters could therefore be defined as concepts in the DD – the alternative definitions of a laboratory parameter could then be analyzed with regard to the different measurement procedures in use. In detail, each concept which is a laboratory parameter would be assigned to a pre-defined category laboratory parameter which defines a template containing a relational template component using the binary relationship measured-by. Each semiformal definition of a concept of the category laboratory parameter would then define DefDependentRelLinks by selecting the measurement procedures (which are also defined as a category and specified as an allowed type of the respective RelationalRole) in use.

Note that DefDependentRelLinks can only be defined in conjunction with a RelationalFilledComp – i.e. they can only be defined for semiformal definitions and only if the template which is used by a semiformal definition contains a RelationalFilledComp that uses the desired Relationship. Experience will show whether this meets the needs of the users. In our opinion it is reasonable not to allow for the definition of relational links at informal definitions since this could lead to redundancy if the meaning expressed by the relational link is also contained in the informal definition. However, if experience shows that it is necessary to allow for the creation of DefDependentRelLinks at informal definitions or independently of RelationalFilledComp at semiformal definitions, the model can easily be extended to accomplish this.

Furthermore, the model does not forbid the usage of the same relationship in both types of links (DefDependentRelLink and DefIndependentRelLink), not even for the same concept. The editors must be aware of this and decide in each case whether it makes sense. Further analysis on the general usefulness and semantics of such occurrences is necessary but beyond the scope of this thesis.

5.5.3 Lexical Relations

In section 2.1, the notions of ambiguity (comprising homonymy and polysemy) and synonymy are introduced. In the following, these general definitions are refined by defining these notions on the basis of the classes in the DD model.

Definition of Synonymy

While the DD model distinguishes between Strings and Terms, this distinction is not made by the general definition of synonymy given in section 2.1 – this definition is based only on the distinction between terms and concepts as defined by the semiotic triangle. It is therefore necessary to analyze whether synonymy in the DD model should be defined between terms or strings.

Defining the synonymy relation between Strings would be wrong because Strings are completely concept-independent, i.e. they do not carry any meaning. Hence, establishing synonymy between Strings would essentially mean only being able to represent absolute synonymy, i.e. the interchangeability of Strings in every context. As mentioned in section 2.1, it is widely accepted that absolute synonymy is very rare (cf. [Fellbaum, 1998; Lyons, 1995]). Strings can have – due to the occurrence of homonymy and polysemy – different meanings depending on the context in which they are used, it thus depends on the meaning or Concept which is to be expressed using the String whether another String can be used to express the same Concept.

Hence, the synonymy relation is modeled between Terms which are concept-dependent as explained in detail in the section on terms (5.3.2). A Concept can be designated by multiple Terms – the set of synonyms for the Concept.

> Two instances of Term are synonymous to each other if they are linked to the same Concept.

A further advantage of defining synonymy between Terms instead of Strings is that it avoids considering lexical variants of the same Term as synonyms (e.g. singular and plural forms). The selection of a specific lexical variant of a term is governed by grammatical rules and is unrelated to the issue of synonymy.

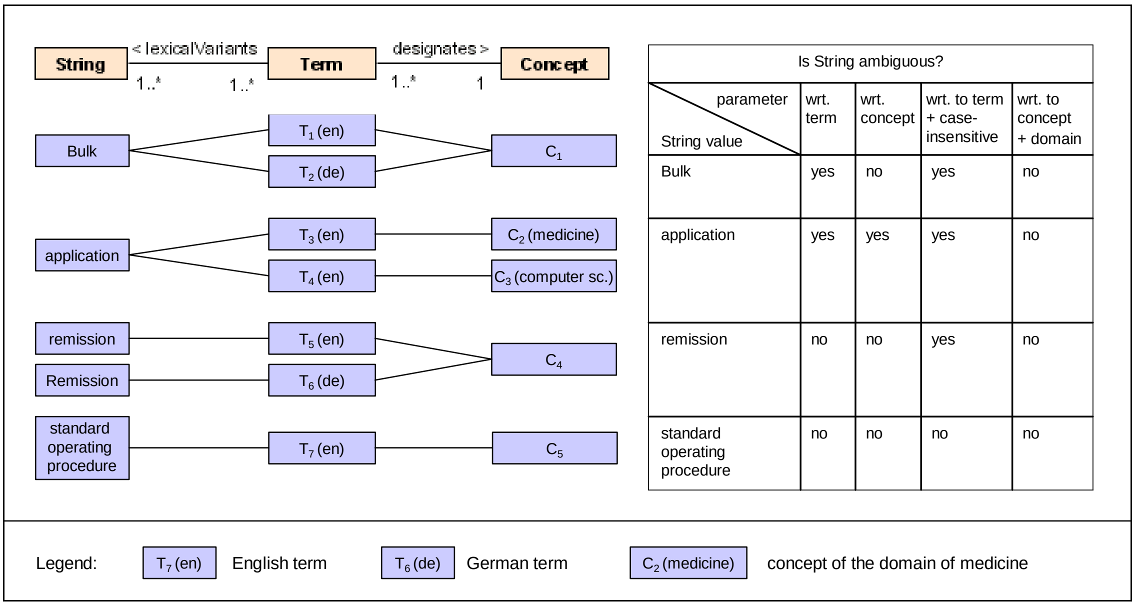
Definition of Ambiguity

Homonymy and polysemy are defined in section 2.1. Since there exists no precise definition of the difference between the two notions, we introduced the more general notion of ambiguity as a term having more than one meaning. As with synonymy, this general definition needs to be adapted to the string-term distinction in the DD model.

Since Terms are linked to exactly one Concept, it is obvious that ambiguity must be defined as a property of Strings. The straightforward solution would consequently be to define a String as ambiguous if it is linked to more than one term. However, there might exist multiple terms for a string although it is linked to only one concept. For example, the same string can be a lexical variant of terms of different languages although all these terms designate the same concept: bulk is a specialist term which is used in English and German and is thus represented in the DD by two distinct terms, one per language. Based on terms only, bulk would be considered as an ambiguous string, but it is unambiguous with respect to the concept to which it refers. Therefore, it appears appropriate to distinguish between ambiguity based on terms and on concepts, respectively:

> A String is *ambiguous with respect to terms* if it is a lexical variant of more than one Term. Alternatively, a String can be considered as *ambiguous with respect to concepts* if it is linked to more than one Concept (indirectly via instances of the class Term).

There are further parameters which may influence the notion of ambiguity. First, strings are case-sensitive and it might be desirable to ignore case when searching for ambiguous strings. For instance, the case-insensitive string remission is ambiguous with respect to terms as illustrated in figure 5.15. Secondly, it could depend on the intended purpose whether a string should be considered as ambiguous if it is linked to concepts which belong to different subject areas (represented in the DD by the class Domain introduced in section 5.7.2). As an example, the string application is ambiguous (in the sense of a software application or in the medical sense of applying medication), but if the domain is known (either computer science or medicine), it can be disambiguated (see figure 5.15).



Is String ambiguous?				
parameter String value	wrt. term	wrt. concept	wrt. to term + case-insensitive	wrt. to concept + domain
Bulk	yes	no	yes	no
application	yes	yes	yes	no
remission	no	no	yes	no
standard operating procedure	no	no	no	no

Figure 5.15: Examples of strings, terms, and concepts illustrating different notions of ambiguity.

These examples show that differently strict definitions of ambiguity are possible. Especially in view of the specification of preferred terms which should be “unambiguous” as explained in section 2.1, it seems useful to let the users choose the concrete notion of unambiguity which should be enforced for preferred terms.

The possible parameters for the definition of ambiguity are not elaborated in greater detail here because various further parameters based on the distinctions available in the DD model are conceivable in principle. However, practical experience is missing which of these are most relevant such that further analysis of the practitioners’ requirements is necessary.

5.6 Communities and Content Areas

Requirement R22 calls for the possibility to define separate content areas which are accessible by different communities. Therefore, the class `ContentArea` is defined which allows for the partitioning of the DD into disjoint parts, i.e. each `ContentArea` contains a set of `Concept`s, and each `Concept` belongs to exactly one `ContentArea`. Together with the concept, all information which is associated to it is included in the respective content area, e.g. the definitions, terms and strings. Note that from this it follows that concepts from different content areas cannot be related – content areas are disjoint and independent parts of the DD.[46]

Some content is independent of content areas; this concerns `Category`, `Relationship`, `RelationalRole`, `Template`, `TemplateComponent`, `Domain`, `Role` (the latter two are introduced in section 5.7.2). These classes are different insofar as it seems very desirable to

[46] Allowing for relations between concepts from different content areas is not straightforward since it introduces problems with regard to access rights (a user may not be authorized to see both of the related concepts).

reuse their instances across content areas – otherwise, for example, generally useful relations like part-of, is-a, etc., would need to be defined in each content area separately.[47]

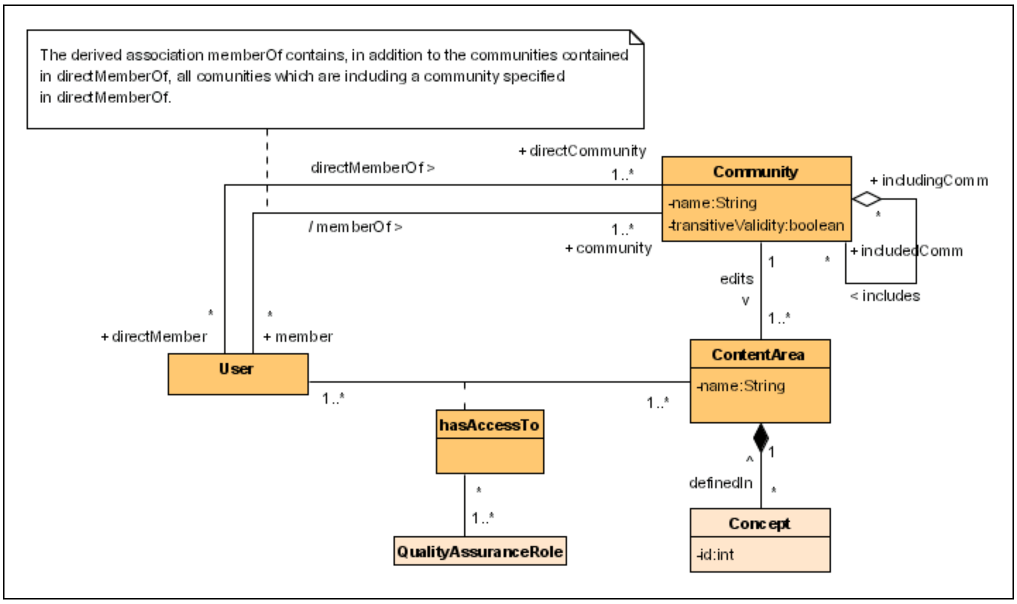


Figure 5.16: Representation of communities and content areas.

Each ContentArea is edited by a Community, which can represent a real organization (e.g. the Competence Network Malignant Lymphomas) or just a group of users or a set of organizations that cooperate in developing a common ContentArea, i.e. they share an interest in jointly developing and harmonizing definitions. The same Community can edit multiple content areas – which has no particular meaning in the DD model – it just allows a Community to create content in separate content areas; the purpose of the different content areas is determined by the community. Experience from the existing first version of the DD shows that a single community, for example, wants to use one content area for testing purposes only and one for developing the content which is to be published.

A User[48] can be a member of multiple communities. Every user has by default at least reading access to all content areas which are edited by the communities of which he or she is a member. Users can further be granted access to other content areas (via hasAccessTo), even if they are not members of the respective editing community. This additional possibility to explicitly grant access to someone who is not a member of the community is included in the DD model in order

[47] However, it is also conceivable that not all categories and relationships defined by one community (= the group of users who edit the content area, cf. the following paragraph) should be available to all communities, e.g. for reasons of security or just because the domains covered by different communities are completely different. Therefore, it might be necessary to introduce further attributes for these classes which restrict the visibility of their instances to certain content areas.

[48] The attributes of the class User are not specified in the UML diagram since user modeling is developed in prallel in a separate project. Briefly, it will be solved by means of an LDAP-based directory service (LDAP = Lightweight Directory Access Protocol, see http://www.ldap.org for further information). Therefore, we describe only the relevant associations of User with regard to the DD model classes.

to avoid each user to whom access to the DD should be granted having to be declared a member of the respective organization. Patients, for instance, can be allowed to read some content area; they are not members of the editing organization, however.

As mentioned in R22, a ContentArea can thus be employed as a virtually separate DD, where content is developed independently of other content areas. Requirement R23, however, proposes to allow for the sharing of content areas in order to support several communities' working together on the shared content area while still maintaining a separate content area for their own content, which is not to be accessible to other communities. In support of this requirement, the DD model allows for the definition of an inclusion hierarchy of communities via the relation includes. This relation establishes a directed, acyclic graph of communities.

The set of members of a community is the union of the members of all included communities plus users which are explicitly defined as a member via the relation memberOf. A user who is an explicit member of a community C_1 is thus a (derived) member of all communities which include C_1, i.e. all communities which are on the upward path in the graph, and consequently has (at least reading) access to all the content areas of these communities. See figure 5.17 for an example of a community hierarchy. In order to be able to share a given content area between communities C_1 and C_2, a community C_5 is defined which includes C_1 and C_2. (The respective content areas for each community are not shown in the figure.) Now, all users of communities C_1 and C_2 have access to the content area of C_5, but with possibly different QualityAssuranceRoles (e.g. editor, moderator, etc.).[49]

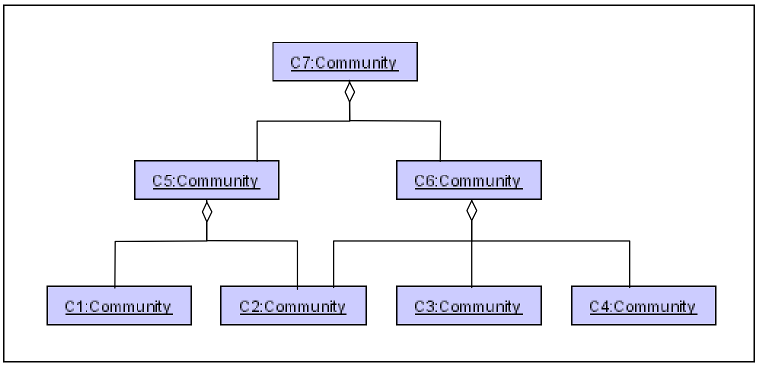


Figure 5.17: Illustration of a community hierarchy.

The inclusion hierarchy of communities is also of importance with regard to the organizationalValidity of an alternative definition. This attribute is introduced in section 5.4.1, and described as being used for indicating the scope of validity of a definition. The purpose of this information is to allow for the selection of all definitions which are valid for an organization and thus to be able to obtain a glossary from the DD containing only the definitions valid for some community. The inclusion hierarchy of communities allows for the specification of a scope of validity which includes several communities. Consider again the example in figure 5.17. C_5 is a community which represents the cooperative work of C_1 and C_2. Definitions created

[49] As already mentioned in section 5.4.1, the quality assurance cycle (including access rights of users, functional rights during the quality assurance workflow) is, like the topic of user modeling, not covered further within this thesis but is being developed in a separate project.

in the content area C_5 can have the value C_5 as their organizationalValidity if the definition is agreed on by C_1 and C_2, i.e. it is valid for both included communities. If, however, both organizations are still using different definitions, they can both enter an alternative definition with organizationalValidity C_1 or C_2, respectively, in order to indicate the scope of validity of the alternative definition. During the harmonization process, they can develop a consensus definition which has C_5 as its organizationalValidity. Thus, it is possible to collaboratively use the same content area in order to be able to compare the various alternative definitions (towards the goal of harmonization) while still being able to select only those definitions which are valid for one of the cooperating communities.

As a refinement of the validity of definitions which is, as described in the previous paragraph, interpreted transitively with regard to included communities, we propose the additional Boolean attribute transitiveValidity at the class Community, which can block this transitive validity for included communities. The motivation is that two types of communities are conceivable: i) communities whose content is mandatory for all included communities, and ii) communities whose content is valid within the defining community but not automatically for all included communities. As an example of the first type, consider the definitions in the Good Clinical Practice [ICH, 1996]. These definitions are valid for all clinical trials, i.e. we could define a community GCP with the content area A_{GCP} which contains all definitions of the GCP. All communities which want to be compliant with the GCP can be defined as included communities (i.e. in the inclusion hierarchy, they are all sub-communities of GCP). Thus, all members of these communities have access to the content defined in A_{GCP} and, due to the inclusion relationship, the definitions in A_{GCP} with organizationalValidity=GCP are also valid for all included communities. An example of the second type is a community C_2 which is cooperating with different communities in various projects, and is thus included in several communities (see figure 5.17 above: C_2 is included in C_5 and C_6). Here, it may, for example, not make sense to have the transitive validity of definitions with organizationalValidity=C_5 because C_2 has to use the definitions valid for C_5 only if it is working in this organizational context. However, if C_2 is working in the context of C_6, it has to use the definitions valid for C_6.

In summary, if for a community C the attribute transitiveValidity is true, a definition with organizationalValidity=C is also valid for all communities C_i which are included in C, i.e. a query to retrieve all valid definitions for an included community C_i will also include definitions with organizationalValidity=C. If the value of transitiveValidity is false, however, the query for definitions valid in C_i would not include definitions valid for C.

The context and view mechanism described in the next section will support the user in selecting only those definitions which are valid in the current working context (cf. view axis V_organizationalValidity defined in section 5.7.1).

5.7 Representation of Contexts and Views

Thus far, it has been specified and explained how concepts are defined in the DD. This section deals with the representation of contexts and views. Note that in this section we deal with situative context only; the role of linguistic context is briefly discussed at the end of section 5.8. In the following, we will therefore use context as a shorthand for situative context where it is unambiguous.

In section 5.7.1 we start to reconsider the general definitions of view and situative context and from these develop concrete definitions for the DD model. Afterwards, section 5.7.2 defines additional model elements which are necessary in order to support the context and view mechanism in the DD.

5.7.1 Definition of Context and View in the DD Model

Let us briefly recapitulate the general definitions of the notions of view and situative context as introduced in section 2.5. Situative context is defined as the sum of any characteristics of the situation in which an utterance occurs. In relation to the DD, ‘utterance’ can be considered to comprise any query to the DD. Context is identified as being important with regard to the disambiguation of words as well as for selecting relevant information (context-based information presentation). Secondly, a view is defined as a projection of a base model containing only entities (or more general: information) relevant to the current “perspective or vantage point” ([OMG, 2003a] p. Glossary-16).

A question that arises from these definitions is how context and view are related, i.e. whether – and if so, how – they influence each other. In analyzing the definitions summarized above, it can be discovered that both definitions are concerned with the relevance of information. However, although the definition of view states that a view includes only information relevant to a specific perspective or vantage point, it does not specify how to determine the relevance of such information. In particular, it does not specify what a “perspective or vantage point” is. Keeping in mind that this definition of view is taken from the UML specification [OMG, 2003a], we interpret this definition such that “perspective” includes, for example, the current task and purpose which influence the necessary level of detail of a UML model as well as the conceptualization itself (cf. reason 2 in section 3.2.1). At this point, a relationship to the notion of situative context can be established. In our opinion, what is called “perspective or vantage point” in the definition of view in the UML specification, is comparable to what is called situative context in other approaches. Hence, we suggest rephrasing the definition of view in order to reflect its relation to situative context:

> A view is a projection of the base model which contains only the content relevant in the current situative context.

In other words, the situative context influences the view insofar as it determines which aspects of the base model are relevant in the current context and thus included in the view.

Having established a general connection between context and view, these notions and their relation must now be concretized with respect to the DD model. In particular, the following questions arise:

1. Does any arbitrary subset of DD content assemble a view or are there additional criteria that characterize and distinguish a view from an arbitrary subset?
2. Which characteristics should be used to represent context in the DD model? This question arises because in practice it is obviously impossible to represent all characteristics of a situation. For the context representation in the DD model, it is therefore necessary to select a manageable number of characteristics which are most important to the selection of views.
3. How exactly does the context influence the view?

The first question addresses the issue of whether any subset of DD content should be regarded as a view. This is certainly not the case. Consider for example an arbitrary subset of the DD containing a few unrelated terms, strings, and concepts. Such an arbitrary subset is useless because it does not consider the relations that exist between these and other instances (e.g. concepts are designated by terms, strings are lexical variants of terms) For example, a single string is of little use if it is isolated from its related term(s) and concept(s). Of course, there may be technical use cases for creating such subsets, but this is not – in our opinion – what should be regarded as a view. Therefore, the above definition of view is refined by adding that a view must be coherent. Concerning the DD, this means that a view does not contain isolated instances (e.g. single concepts, single definitions or single terms), but selects the relevant instances according to certain criteria – namely the view axes described below – and adds further related contents. The general idea of adding related contents is to start from the instances selected by a view axis and to include instances of certain other model classes which can be reached via certain model associations (cf. Appendix A). As an example, if a definition is included in the view, the concept being defined as well as the designating terms together with the strings which are their lexical variants need to be included as well.

After reconsideration of the classes of the DD model presented thus far, as well as the requirements presented in section 3.2), we identify eight **view axes** (see below) which lead to useful subsets of DD content. Each view axis affects a different aspect of the DD model, i.e. the view axes are independent of each other and can be applied in arbitrary combinations. A **view** on the DD model is the combined result of applying one or more view axes on the DD.

The context's influence on the view is represented as follows. Situative context is represented by a set of **context properties**, and each view axis depends on one of these context properties[50]. Thus, each view axis can be regarded as a function which takes the DD content as its input and depends on a context property as its parameter. The result of this function is a view (i.e. a subset of the content taken as input). If multiple view axes are applied, they are computed one after another where each subsequent view axis is computed on the basis of the result of the previously applied view axis. The basis of each view computation is that part of the DD to which the user has access (i.e. a set of content areas). If no view axis is applied, the user can thus see all content

[50] Except for one view axis (`V_organizationalValidity`) which requires two similar context properties.

to which he or she has access. Regarding the duration of views, we suggest that once a view is computed, it should be retained for all subsequent queries and searches in the DD until the user changes any context properties or selects different view axes causing the view to be recomputed.

The following eight view axes are proposed: `V_ContentArea`, `V_organizational-Validity`, `V_geographicalValidity`, `V_RoleRelevance`, `V_degreeOfDifficulty`, `V_Category`, `V_Domain`, `V_Language`. The motivation for each view axis, the context properties it depends on (given in brackets), as well as its effect on the DD content is described in the following.

Note that some of the DD model elements which are needed for the computation of some view axes are not introduced yet because their definition is motivated by the respective view axes and therefore cannot be defined before introducing the view mechanism. This concerns the view axes `V_RoleRelevance`, `V_degreeOfDifficulty`, and `V_Domain`, which motivate for example the definition of the classes `Domain`, `Role` as well as the attribute `degreeOfDifficulty` for terms and definitions. These model elements are defined in section 5.7.2.

V_ContentArea(C_Areas)

As described in section 5.6, the total set of concepts defined in the DD is partitioned into `ContentArea`s, where each concept is defined in exactly one content area, and each user of the DD has access to a number of content areas. The view axis `V_ContentArea` allows for creating a view in which only a subset of those content areas to which the user has access is visible and the other content areas are hidden (and, consequently, all the content defined therein). The context property `C_Areas` specifies the set of content areas which are visible in the view.

V_organizationalValidity(C_Communities, C_OrgValString), V_geographicalValidity(C_GeogrAreas)

These two view axes are motivated by the purposes of the DD discussed in section 5.4.1. One of these purposes is that the DD is to be used as a glossary of valid definitions, i.e. users should be able to ignore alternative definitions which are invalid within their current working context. The view axes `V_organizationalValidity` and `V_geographicalValidity` support the latter purpose. Given a set of organizational (specified by `C_Communities`, `C_OrgValString`[51]) or geographical contexts (specified by `C_GeogrAreas`), respectively, these view axes include only those definitions in the view which are valid within the specified organizational / geographical context. Invalid definitions are thus hidden in this view, but remain available for purposes of analysis.

[51] As elaborated in section 5.4.1, the organizational scope of validity can be specified by the association `organizationalValidity` pointing to a set of communities and / or by filling the attribute `orgValidityComment` with a textual comment about the validity. Accordingly, two context properties are needed for the view axis `V_organizationalValidity`, which actually represent the same “dimension” of context, i.e. organizational context.

V_RoleRelevance(C_AreaRoles)

In section 3.2.1, the reasons for the occurrence of differences in definitions are analyzed. Reason 3 states that differences in definition occur due to the interdisciplinary character of complex domains like that of clinical trials. Different groups of persons have different information needs due to their different tasks and knowledge. Section 5.7.2 describes how roles and role relevance of content are represented. The view axis `V_RoleRelevance` allows for the selection of information which is relevant for specific roles (specified by `C_AreaRoles`) while hiding irrelevant information. Briefly, this view axis affects the selection of concepts or parts of their definitions.

V_degreeOfDifficulty(C_DomainCompetences)

`V_degreeOfDifficulty` is, like the previous view axis, also motivated by reason 3 which mentions the interdisciplinary character of complex domains. However, `V_degreeOf-Difficulty` addresses the issue of different groups of persons having different competences in domains and thus needing information at different degrees of difficulty. Based on a set of competences in domains (`C_DomainCompetences`, e.g. `biometry`; see section 5.7.2), this view axis therefore allows for viewing the content of the DD at the degree of difficulty appropriate to users' competences. Section 5.7.2 describes how content at different degrees of difficulty is represented in the DD model. In short, this view axis affects the selection of terms (layman or expert terms) as well as definitions (layman or expert definitions).

V_Category(C_Categories, showFullDef)

As described in the section on categories and semiformal definitions (5.4.2), a category groups similar concepts and can define a template, the purpose of which is to standardize the semiformal definitions of concepts of that category. The question of whether concepts are similar and should thus be assigned to the same category is a classification problem, which depends on the point of view as explained in reason 2 in section 3.2.1. Thus, in order to allow for multiple purposes and points of view, the DD model allows for a concept to belong to multiple categories. Consequently, a semiformal definition of a concept belonging to multiple categories can consist of multiple parts (`FilledTemplate`) where each part fills the template of one category. Each of these parts provides a partial description of the concept from one point of view.

The purpose of the view axis `V_Category` is therefore to display view-specific definitions of concepts: Given a set of categories (`C_Categories`), the view includes only concepts of these categories, and – if the additional parameter `showFullDef` is set to `false` – displays only those parts of the semiformal definitions which belong to the respective category templates.

V_Domain(C_Domains)

This view axis restricts the view to concepts of certain subject areas represented by the class `Domain` (cf. section 5.7.2 on the representation of domains). Only those concepts which belong to a set of domains specified in `C_Domains` are included in the view.

V_Language(C_Language)

As described in the previous sections, the DD contains multilingual content, i.e. terms and definitions in several languages. The view axis V_Language allows the user to restrict the view to a certain language, i.e. only terms and definitions of a specified language (C_Language) are included while content in other languages is hidden.

An overview of the view axes and their dependence on context properties as well as the specification regarding how each view axis is computed from the respective context property is given in Appendix A.

The remaining questions are how the current situative context of a user is determined (i.e. how are the values of context properties set) and which conditions cause a certain view axis to be applied. In general, two approaches are conceivable. The first possible approach is to determine the user's context and most appropriate view axes automatically on the basis of certain user data. The second possible approach is to let the user set the context properties manually and choose the desired view axes which are to be applied.

Since there are currently no methods and user data available on the basis of which the appropriate view axes and context could be determined automatically[52], we choose the second possibility. Of course, an implementation of the context and view mechanism should allow users to select view axes which are to be applied by default, as well as to specify default values for those context properties required for computing the view axes.

However, one step towards the first approach of automatic user evaluation is that some known properties of users are employed as default values of context properties and thus necessitate a manual setting only in cases where a different context property setting is required. For example, each user is known to be a member of a set of communities. This set is used as the default value for the context property C_Communities. If a user applies the view axis V_organizationalValidity, she will thus see by default all those definitions which are valid in the communities she is a member of. As discussed in the section on communities (5.6), it may, however, be the case that a user needs to accomplish a task in which only definitions valid for some specific community may be used. In other words, the current organizational context of the user is some specific community C_i, which the user can set as the value of the context property C_Communities. When applying the view axis V_organizationalValidity, the user will consequently only see definitions valid in the current organizational context C_i. Details on user properties and their usage as default values for context properties is also given in Appendix A.

This section concludes with a summary of the answers given to the three questions raised at the beginning of this section, which are concerned with the adoption of the generic definitions of view and context to the DD model. The first question asked for criteria that characterize and distinguish a view from an arbitrary subset of DD content. This has been answered by the coherence criterion as well as the eight view axes which determine the selection of content. The second question regarding the concrete representation of (situative) context in the DD model has

[52] Context could, for instance, be determined on the basis of external applications which access the DD.

been resolved by specifying a set of context properties. The third and final question about the actual influence of context on view is addressed by employing context properties as parameters of view axes and by specifying the view computation in detail (Appendix A).

5.7.2 Model Elements Supporting the Context and View Mechanism

The context and view mechanism necessitates the definition of a few additional model elements which are described in the following. In particular, this concerns the representation of domains, different degrees of difficulty as well as relevance of content. Figure 5.18 provides an overview of the respective classes and associations.

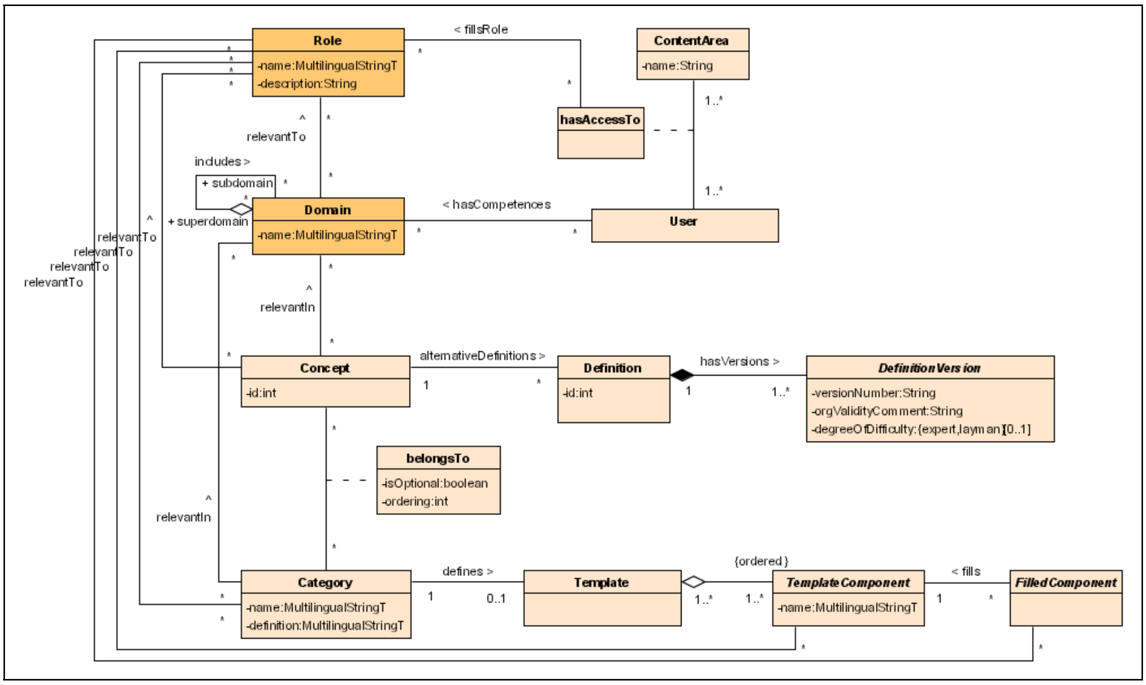


Figure 5.18: Model classes and associations for representing contexts.

Representation of Domains

Related to the representation of domains are the class `Domain` as well as the two associations `relevantIn` between `Concept` and `Domain` or `Category` and `Domain` respectively. Instances of the class `Domain` represent subject areas, e.g. medicine, computer science, biometry, or more specific domains like oncology, or paediatric oncology. Concepts can be specified as being relevant in certain domains via the associations `relevantIn` – either individually for a concept (by using the association between `Concept` and `Domain`) or for all concepts of a certain category at once (by using the association between `Category` and `Domain`). Thus, domains organize (or modularize) the set of concepts into overlapping subject areas.

It is acknowledged that representing domains is a complex topic in itself and that the definition and demarcation of subject areas is difficult. We therefore strive for a pragmatic approach which is driven by the following aims (each explained below):

1. context-based information presentation on the basis of domain context
2. disambiguation of strings
3. representation of content at different degrees of difficulty

Firstly, the organization of concepts into subject areas allows for a focus on the concepts of certain subject areas while ignoring others. Note that domain is only one facet of context-based information presentation (requested by requirement R14); in fact, each view axis describes a different facet that contributes to achieving this aim. Secondly, knowing the domain context in which an ambiguous string is used may make disambiguation possible, namely if the several concepts which could be designated by the string are relevant in different domains. These two goals are achieved by the view axis `V_Domain` (cf. section 5.7.1) because it allows for creating a view that focuses on certain domains (specified by the context property `C_Domains`), excluding concepts not relevant in the domain. Consequently, if a search is conducted for a string which is ambiguous in the unrestricted view, this string could be unambiguous in the domain-specific view since the latter may include fewer concepts and thus affect the mapping from the string to concepts.

The third and final objective supported by the domain representation is that the meaning of concepts can be described at different degrees of difficulty as described later in this section (in short, if a user is known to be knowledgeable in the subject area of a concept, he is presented with the expert definitions and terms associated with the concept, but with layman definition and terms otherwise).

Domains can be organized in a polyhierarchy via the association `include` which is understood transitively, meaning that the concepts relevant in the “superdomain”, i.e. the domain which includes other domains, are relevant in all subdomains as well but not the other way around. For example, the domain `oncology` would be a subdomain of `medicine`. In particular, concepts of the superdomain are more generic and may be necessary to understand and define the more specific concepts in the subdomains. However, the latter aspect has not yet been addressed and represented explicitly in the DD because it requires further investigation of related difficult issues like that of defining complex concepts (usually, there is a correlation that domain-specific concepts are defined as complex concepts on the basis of more generic elementary concepts; cf. complex concepts in GALEN [Rogers, J. E., Rector, 1999]).

Furthermore, a concept can be specified as being relevant in multiple domains. Although this might seem redundant against the background that a domain can have multiple superdomains, it is allowed in order to avoid having to create a new subdomain for all cases where a concept is relevant in multiple domains. As a rule, a new subdomain of two domains should be created only if a significant number of concepts is considered relevant in both domains.

On the basis of this domain representation we can conclude that an application of the view axis `V_Domain` results in a view which contains only concepts relevant in the domains specified directly in `C_Domains` or their superdomains.

As mentioned at the beginning, the issue of representing domains is difficult, in particular because it is impossible to define precise borderlines between subject areas. Therefore, it is recommended that one avoid defining too many domains resulting in an unmanageable level of granularity. The more domains are defined, the more difficult their demarcation becomes, as does the decision of which domains a concept is relevant in. As a guideline, a new domain should be defined only if it is of benefit with regard to at least one of the three aims described above. Avoiding domains which are too fine-grained is furthermore important to avoid blurring the distinction between domains and concepts.

Representation of Content at Different Degrees of Difficulty

In section 3.2.1 the need for content at different degrees of difficulty is explained (reason 3: Interdisciplinary Character of Complex Domains). The main issue described therein is that complex domains like that of clinical trials include knowledge from a wide range of specialty areas, and individual users cannot be expected to be knowledgeable in all of these areas. Instead, while they may understand and require expert knowledge in some domains, they may need less specific and detailed information in others.

This issue is addressed in the DD through the attribute `degreeOfDifficulty` at the classes `Term` and `DefinitionVersion`, as well as the association `hasCompetences` between `User` and `Domain`. The association `hasCompetences` represents that a user is knowledgeable in a given set of domains (= “expert”), and layman in all other domains (i.e. domains which are neither selected directly nor are a superdomain of a directly selected domain; recall that superdomains contain the more generic knowledge necessary to understand subdomains). Using the domain information of a concept (a concept is relevant in a set of domains, see above), it is thus possible to divide the complete set of concepts into two subsets, one containing all concepts which belong to the user’s expert knowledge, and another containing the concepts of the user’s layman knowledge, respectively.

Having represented the user’s competence in a set of domains, the question arises in which way information about a concept should be represented in order to meet the user’s needs concerning the difficulty of content. As identified in the requirements section, laymen need different terms as well as different style and detail of definitions. Therefore, the attribute `degreeOfDifficulty` at the classes `Term` and `DefinitionVersion` may specify that a term or definition is appropriate for either an `expert` or `layman` in the concept’s domain[53]. Specifying this value is optional – an unspecified `degreeOfDifficulty` is interpreted as being relevant for both experts and laymen.

Applying the view axis `V_degreeOfDifficulty` results in a view which contains only terms and definitions which match the competences specified in `C_DomainCompetences`. In particular, concepts are excluded from views which, when applying the view axis, do not have any term and definition anymore (according to the coherence of views, i.e. no isolated instances are contained in a view; cf. section 5.7.1). This allows for excluding whole concepts which may

[53] As described in the section on `Domain`, assigning a concept to one or more domains is optional. It must therefore be ensured that the degree of difficulty can only be specified if the concept is assigned to at least one domain.

be, for example, too specific for a layman of a domain or too unspecific and thus uninteresting for an expert of the domain.

Note that it is in the responsibility of the editors of terms and definitions to avoid incoherent cases, though support should be provided as far as possible by the user interface. As an example, if a layman definition exists for a concept but a term is missing in the layman view (or, the reverse: a layman term exists but no layman definition is available), a warning should be produced asking the editors to add a suitable term (or definition). If no separate layman term (or definition, respectively) is to be developed, the existing term or definition should leave the `degreeOfDifficulty` unspecified, which is interpreted as being appropriate for both experts and laymen.

We are aware that this approach towards representing content at different degrees of difficulty presented above introduces a number of simplifications. In particular, the distinction between experts and laymen as well as the representation of domains is coarse. However, allowing for a finer grading of levels of difficulty requires much more effort in creating content, since each definition should ideally be available at each degree of difficulty. In our opinion, the benefit of a finer grading would be minimal, if any, and hence does not justify the considerably higher effort which would be required for developing content.

We claim that our approach provides a good compromise between being feasible while still being capable of representing the difference between layman and expert terms and definitions, which is of concrete practical relevance, for example in the development of patient information. As mentioned above, the approach is even capable of hiding whole concepts in a difficulty-adapted view which are irrelevant with respect to the user’s competence. A more detailed representation and discussion of this topic is beyond the scope of this thesis, however.

Finally, the representation of content at different degrees of difficulty should be compared to the translation of definitions into different languages (e.g. English, German). As described in section 5.4.2, it is possible to translate each `DefinitionVersion` into several languages which ensures that definitions which are substantially the same are not considered as alternative definitions in the harmonization process. Although the issue of different degrees of difficulty seems to be similar to “translating” a definition, it is in fact represented as different alternative definitions. One reason for this decision is that the format may be different, i.e. a layman definition could be an informal definition and the expert definition a semiformal one. As explained in section 5.4.2, however, the translation of definitions partly depends on the format. Hence, the expert and layman definitions are represented as different alternative definitions. It may be necessary, though, to introduce an association by which definition versions can be linked that are essentially the same definition with regard to content but formulated at different degrees of difficulty.

Representation of Role Relevance

As explained in section 3.2.1 regarding the interdisciplinary character of complex domains (reason 3), different users have different information needs due to their task and knowledge. While the latter aspect regarding the knowledge of users is addressed by degree of difficulty (described in the previous section), this section is concerned with task- or role-related relevance of information. Two questions are addressed in the following. Firstly, we need to specify what is meant by the notion of task or role, and secondly, the effect of roles on selecting content for the view axis `V_RoleRelevance` needs to be defined.

The answer to the first question is not easy, since it is another vague notion which is related to similar notions like role, functionality, and responsibility. For example, a human may play the role of a documentation staff member in a clinical trial, which entails tasks and responsibilities regarding the documentation of study results. Since these notions cannot be elaborated in detail herein, we aim – as with other model elements related to view and context representation – at a pragmatic approach which allows for making distinctions relevant to the DD.

Our approach to representing role relevance is explained below according to the following two of the four examples given in section 3.2.1, reason 3:

> `Biometricians are especially interested in biometrical concepts (e.g. <Karnofsky Index>), biometrical methods that may be used for the statistical analysis of certain concepts as well as information about statistical problems related to concepts.`
>
> `Documentation staff are interested in any issues regarding documentation, e.g. the representation of a concept on a case report form.`

We introduce the class `Role` (according to the abovementioned intuitive understanding of role), which could have as its instances for example `biometrician`, or `documentation staff`. (Of course, if the DD is applied in a different domain than that of clinical trials, it would contain different roles. For this reason, we do not specify a certain set of roles since the model of the DD should remain domain-independent.)

From the two examples above and the model classes contained in the DD, we can derive the following possible cases:

1. Concepts can be relevant to specific roles, i.e. all information about a concept is relevant and displayed in a role-specific view (e.g. `<Karnofsky Index>` is relevant to a `biometrician` but (generally) not to other roles).
2. Only certain parts of the definition of a concept are relevant to specific roles, while the rest of a definition is irrelevant (e.g. `issues regarding documentation` of concepts is relevant to `documentation staff`). (Note that this is only possible for semiformal definitions which are structured into parts, namely filled template components.)

The first case is represented by the association `relevantTo` between `Concept` and `Role` (see figure 5.18). Further, in order to allow all concepts of a category or of a domain to be collectively assigned as being relevant to a role, the association `relevantTo` is also defined between `Category` and `Role` as well as between `Domain` and `Role`.

The second case is represented analogously, i.e. an association `relevantTo` is defined between `FilledComponent` and `Role`. Here, the additional association between `TemplateComponent` and `Role` serves the purpose of providing a higher level where relevance can be specified.

The details regarding how the view axis `V_RoleRelevance` exploits the information represented by the class `Role` and the several `relevantTo` associations are given in Appendix A. In summary, informal definitions are included on the basis of the concept's relevance, i.e. if the concept is relevant to a role, each informal definition is included in the view. Semiformal definitions are handled differently, because each of their definition components can have a different role relevance: a `FilledComponent` is included in the view either if it is explicitly specified as relevant or, if no role relevance is specified for the component, it is included on the basis of the concept's relevance. In other words, the concept's relevance is the default for all definition components, but can be overwritten at each component separately because a role relevance specified explicitly for a component has priority over the concept's overall relevance.

Another possible approach would have been to always include the whole semiformal definition on the basis of the concept's relevance and use the definition component level only for specifying additional roles. Not having chosen this approach is motivated by the idea that there is a conceptual difference between both levels. In fact, the examples given thus far to motivate the representation of role relevance at the level of definition components are all merely technical but not related directly to a concept's definition. Thus, in our approach, role relevance specified at the definition component level indicates that this component is not part of the definition but is rather additional (technical) information related to the concept (cf. requirement R8 about representing additional knowledge), for example the representation of a concept in a database, on a case report form, etc.

5.8 Searching for Concepts by Strings

After having described all elements of the DD model, it is now possible to explain how a search for concepts is carried out. The DD needs to provide a search function that allows users to navigate from strings to terms and concepts. As a reminder, figure 5.19 shows the classes and associations relevant for understanding the explanations below.

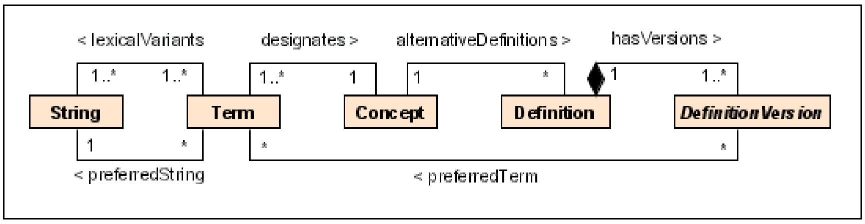


Figure 5.19: Excerpt from the DD model providing an overview of the classes and associations relevant to the resolution of strings to concepts.

The difficulty in resolving strings to terms and concepts is the occurrence of ambiguity, i.e. a string can have more than one meaning which becomes manifested in the DD as a string that is a lexical variant of multiple terms, and thus (indirectly via term) designates possibly more than one concept. It is therefore desirable to establish a method for sense disambiguation of strings which supports the user who is searching for a concept by entering a string in selecting the intended concept from the set of concepts which come into question.

The context and view mechanism can support sense disambiguation in the DD because a view contains only a subset of concepts and terms, namely those which are relevant in some context. As a consequence, context (more precisely: a specific setting of context properties) may reduce the number of terms and concepts to which a string can be mapped. Hence, if a search for a string returns more than one concept, the DD could determine the contexts of these concepts, and derive contexts in which the string is unambiguous (i.e. mapped to exactly one concept) or at least less ambiguous (i.e. mapped to fewer concepts). Then these contexts could be presented to the user who could select the one which is most appropriate. For example, the domains of two concepts which come into question may be different so that the string can be disambiguated by choosing a particular domain. It is of course also possible that the user specifies the current context before conducting the search.

In addition to the context properties defined in section 5.7.1 which restrict the search to a subset of DD content, three further parameters can be used in mapping strings to terms or concepts. Firstly, one can choose whether the search is to be conducted in a case-sensitive or case-insensitive manner. Obviously, a case-sensitive search will generally result in a smaller number of terms. Secondly, the association `preferredString` could be exploited in order to narrow a search by including only those terms (and thus the respective concepts) in the search result of which the string is a preferred string. Analogously, the third search option is to exploit the association `preferredTerm`, which includes those terms which are the preferred term for at least one definition.

As a last remark concerning sense disambiguation, it should be mentioned that according to the above description, the process of sense disambiguation is based on a definite selection of concepts: starting from the set of all concepts that a string can designate in a given context, concepts are either included or excluded on the basis of certain parameters. An interesting enhancement would be to instead perform a ranking of the concepts, where the concept which is most likely intended by a string is presented with higher precedence than others. For example, linguistic context could assist in this matter. In section 2.5, it was defined as the surrounding text of a word or utterance. Thus, with regard to searching in the DD, a query might consist of multiple strings in which the linguistic context of each of the strings is constituted by the other strings. Utilizing linguistic context means that each single string is not disambiguated in isolation as described above but rather in relation to the other strings. This can be done by analyzing the sets of concepts which come into question for each single string for similarities with regard to the contexts to which they are assigned. Consider, for example, a query consisting of two strings, one of which is unambiguous. The contexts (e.g. domain, category, etc.) of the possible concepts of the second string can be compared to those derivable from the first string. Those concepts which exhibit the greatest similarities in context are more likely to be the intended targets of the search.

6 Discussion of the Model

6.1 Evaluation against the Requirements

In this section, the model presented in chapter 5 is evaluated against the requirements collected in chapter 3. Table 6.1 provides a summary of the requirements and refers to the respective sections of the model chapter which address the requirement or provides a brief comment describing the extent to which the requirement has been fulfilled. If a requirement is addressed in multiple sections, comments in parentheses behind the section numbers indicate which part of the requirement is addressed in the respective section

As can be observed from this table, many requirements are fulfilled directly by certain model elements. Issues regarding these model elements are discussed in the subsequent sections of this chapter. Requirements which are not fulfilled completely are discussed in the remainder of this section.

The only requirement which remains unaddressed within this thesis is the representation of rules (R7), since it is a complex topic in its own right. A formalism for representing rules needs to be developed, in order to allow for automatic interpretation and processing of rules. Further, as mentioned in the requirements section, the functional requirements are collected rather for the sake of completeness as well as for the purpose of discovering any additional model elements required, as is the case with `ContentArea` and `Community`.

Three requirements are fulfilled in part only. Firstly, this concerns the part of R1 calling for simple layout and formatting options of definitions. As currently specified in the DD model, a definition is stored either in simple text format (`InformalDefVersion`, attribute `content` is of data type 'string') or in a structured format (`SemiformalDefVersion`, consisting of `FilledComponents`, which are of primitive data types), both without any special formatting. How formatting is implemented is more a technical than an analytical question and has therefore been excluded from the specification of the DD model within this thesis.

Further, relating documents to concept definitions as "additional knowledge" (cf. R8) has hardly been covered by the DD model thus far. From the modeling point of view, this requirement is rather easy to achieve, because, as suggested by R8, concepts could just be used as indexes to the documents. The technical side is more difficult, i.e. how documents are stored or referenced in the DD. Additionally, one should be aware of the fact that the "pure" definition of a concept may not be clearly separable from additional knowledge about a concept. For instance, as described in the section on role relevance (5.7.2), definitions may also contain knowledge related to the statistical analysis of concepts, or about their documentation, which could actually be regarded as additional knowledge related to a concept rather than part of its definition. The distinction is vague, however.

Thirdly, versioning (cf. R10) is addressed only in part. The model contains the necessary model element `DefinitionVersion`, but intentionally does not address the quality assurance cycle, since this is being developed in parallel to this thesis by project members. Further, we have chosen not to include technical details like information about the date the definition version was created, and which user created it, etc., in order not to overload the model with technical details which detract from the more substantial model aspects.

	Requirement	Fulfillment of requirement
Description of concepts (section 3.1)	R1 Definitions	5.3.1, 5.4; layout not addressed
	R2 Templates	5.4.2
	R3 Synonyms	5.5.3
	R4 Abbreviations	5.3.2
	R5 Multilinguality	5.3.2 (terms), 5.4.2 (definitions)
	R6 Relations	5.5
	R7 Rules	–
	R8 Additional Knowledge	Including documents with additional knowledge is not addressed in this thesis. However, the borderline between definition and additional knowledge is vague, thus some additional knowledge will be entered as part of definitions, possibly marked as relevant to certain roles (e.g. information regarding statistical analyses relevant to biometricians, cf. 5.7.2).
	R9 Classifications	Can be represented through ternary relationships (cf. 5.5.1).
	R10 Versioning	Partly covered by `DefinitionVersion` (cf. 5.4.1), but sophisticated versioning and quality assurance cycle is not addressed within this thesis.
	R11 Semiformal Content	5.4.2 (semiformal definitions), 5.5 (relations)
Alternative def., views and contexts (3.2)	R12 Representation of Multiple Alternative Definitions	5.4
	R13 Explicit Representation of Purpose-Specific Aspects	5.4.1 (organizational and geographical scope of validity), 5.7 (representation of contexts and views)
	R14 Context-Based Information Presentation	5.7
	R15 Separation of Different Hierarchical Views	Supported by expressiveness of relations (explicit relationships, which indicate the meaning of a relational link between concepts). Correct usage of relations is however an issue of content, not of the model. A methodology for the development of untangled hierarchies is likewise beyond the scope of this thesis.
	R16 Support for the Harmonization of Alternative Definitions	quality assurance cycle envisioned, but not covered within this thesis (cf. 5.4.1)
Functional req. (3.3)	R17 – R21, R24 – R27	Functions to edit the DD content (R18) are partly covered by the prototypical implementation (cf. chapter 7). The remaining functional requirements are recorded for the sake of completeness, however, but their implementation is not part of this thesis.
	R22 Separate content areas	5.6 (`ContentArea`)
	R23 Sharing of content areas	5.6 (inclusion hierarchy of communities)

Table 6.1: Overview of the requirements and their fulfillment within this thesis.

Finally, the fulfillment of the requirement to include existing classifications (e.g. the `WHO classification of Malignant Lymphoma`, cf. R9) needs to be explained. This has not been solved by introducing any specific model element, but can be expressed in the DD model by employing relationships. In detail, we suggest creating a category `Classification`, as well as a separate concept for each classification, and assign these concepts to the category. At this point, there are two possibilities for defining a classification. On the one hand, it is possible to enter it in textual format, i.e. as a definition. The second option is to create the concepts of the classification in the DD as well, and to link these concepts by a specifically defined relationship.

For example, one could define a ternary relationship with the relational roles `superconcept`, `subconcept`, `classification`, where the latter allows only concepts as fillers which belong to the category `classification`.[54] In general, the latter option is preferable since it makes the concepts which are part of the classification explicit to the DD, allowing for mappings between different classifications represented in the DD explicitly through the second option.

We acknowledge though that the DD cannot solve the common problem of determining semantic similarity between concepts of different classifications, i.e. whether it is appropriate to map concepts from two classifications to one concept in the DD. This is precisely the problem that the UMLS is concerned with (cf. section 4.1.4). However, the advantage of the DD is that if two concepts from different classifications are mapped to one concept in the DD, the subtle differences in meaning that often exist can be made explicit via two corresponding alternative definitions, which reference the respective classification as their source.

6.2 Reconsidering the Proposed Solution

The proposed solution in section 5.1 outlines the ideal separation of: i) differences in definitions caused by differences in content from differences caused by ii) translation to multiple languages or iii) by different points of view. The suggested model solution is described as representing the first kind of difference by alternative definitions and the second and third type by consistent variants of the same definition. In this section, we examine the extent to which this ideal solution is achieved in the DD model.

Table 6.2 below extends table 5.1, given in section 5.1, in which the reasons for differences are listed and classified according to the kind of differences (content, point of view, language). Table 6.2 adds a third column which complements the reasons with a brief summary of the DD model elements contributing to the representation of the respective differences.

[54] It is also possible to define binary relationships separately for each classification. The expressiveness is the same, but obviously leads to a higher number of relationships. Hence, we prefer the ternary relationship.

Reasons for the occurrence of differences (section 3.2.1)	Kind of difference	Representation in the DD
Reason 1: Disagreement due to Lacking Evidence	content	alternative definitions[55]
Reason 2: Different Conceptualizations according to Different Purposes (Different Classification, Different Is-A Granularity, Different Part-Of Granularity)	point of view	Different classification is partly addressed by the fact that one concept may `belongTo` multiple categories and allows for different views on the same alternative definition by displaying only the part which fills the category template (`V_Category`) Different is-a and part-of granularity are not addressed within this thesis – and will thus occur as alternative definitions.
Reason 3: Interdisciplinary Character of Complex Domains	point of view, language	Concepts or parts of their definitions are `relevantTo` certain `Roles`. Terms and definitions may indicate a certain `degreeOfDifficulty`; the same definition at different degrees of difficulty occurs as alternative definitions.
Reason 4: Different Scientific Requirements	content	alternative definitions
Reason 5: Different Specificity of Domain Context	content	alternative definitions
Reason 6: Organizational Differences	content	The organizational scope of validity of an alternative definition can be indicated by `organizationalValidity`.
Reason 7: Different Geographical Location	content	The geographical scope of validity of an alternative definition can be indicated by `geographicalValidity` .
Reason 8: Missing Standards	content	alternative definitions
Reason 9: Linguistic Differences	language	Different formulations of a definition within one language occur as alternative definitions. Translations of the same definition to multiple languages are represented as one `DefinitionVersion` which exists in multiple languages.

Table 6.2: Overview of DD model elements for the representation of differences in definitions.

In general, it can be observed that the model elements listed in table 6.2 and the view mechanism[56] of the DD provide support for separating alternative definitions from their consistent variants. However, there are exceptions from the ideal separation, namely in the case of degrees of difficulty which are represented as alternative definitions (cf. section 5.7.2). The DD does allow the degree of difficulty of a definition to be indicated, however, in order to make explicit the reason why another alternative definition exists. Another exception is that different formulations of a definition within one language (reason 9) occur as alternative definitions. We have not considered it useful to introduce some special representation for this case, since we expect that this is rather rare in contrast to translations of a definition. Additionally, while having translations of the same definition is desirable, different formulations within one

[55] abbreviation for: `alternativeDefinitions` of the same `Concept`

[56] The relation between the view mechanism and the different points of view on one alternative definition is discussed in the next section.

language should be avoided (except for different degrees of difficulty, which are represented specifically). Hence, different formulations which are not justified are alternative definitions and should be harmonized.

Further, because the representation of different points of view is a very complex topic (especially reasons 2 and 3), it cannot be covered in all its facets by the DD model. The model elements listed next to reasons 2 and 3 can only provide support for allowing consistent variants of the same definition to be represented, but because of their complexity it is still likely that differences which are actually due to these reasons will be entered as alternative definitions. In any case, as mentioned in the requirements analysis, it may be difficult even for experts to assess whether two definitions differ with regard to content, language or point of view. This problem becomes even harder due to the imprecision of many definitions.

In summary, despite few shortcomings of the DD model with regard to the ideal separation, the model allows for considerable support in this matter, especially in the case of translating definitions to different languages and providing different points of view on the same definition. This constitutes a major improvement as compared to the existing version 1 of the DD and the Onto-Builder (cf. section 1.4). Having adopted the term-concept distinction likewise contributes to a correct representation of alternative definitions, because it does not mix alternative definitions which refer to different meanings of a term, and on the other hand allows definitions to be compared even if they use synonymous terms.

6.3 Contexts and Views

The understanding of contexts and views can be summarized such that a view is the projection of a base model containing only the content relevant in the current context, whereas context is treated in the sense of situative context comprising the characteristics of the current situation (see 5.7.1). Context is modeled by a set of context properties, and a view is a coherent subset of DD content which is computed according to several view axes. Each view axis selects content on the basis of a certain criterion (e.g. `V_Category`, or `V_Domain`) and depends on some context properties as its parameters (e.g. `C_Domains` specifies the domain(s) of the current context – with the effect that concepts of these domains are included in the view while others are excluded).

It should be pointed out that in the DD model two different “base models” are conceivable, both of which play a role in the view mechanism developed. On the one hand, as recapitulated above, the DD content constitutes the base model for the DD view mechanism. On the other hand, the idea in the model approach was that the same definition can have consistent variants, i.e. it can be translated into different languages or described from different points of view (cf. the previous section on a review and discussion of this issue). Hence, with regard to the different points of view on one definition, a single definition can be regarded as a base model from which certain parts (i.e. definition components) are selected. This is implemented by the view axes `V_Category` and `V_Domain`, which are capable of hiding definition components irrelevant in the current context. Thus, the notion of point of view applied to definitions is included in the broader notion of a view applied to the overall DD content.

The understanding of views as a subset of a base model can be observed in other systems as well. For example, UMLS allows for different views by excluding selected source vocabularies (see section 4.1.4). Further, GALEN provides a set of filters by which only part of the Common

Reference Model is visible (see 4.1.5). However, GALEN provides additional support for multiple views in that it allows for multiple hierarchies which can be inferred from the concept definitions. Such sophisticated, automated support of different views is beyond the scope of this thesis since it requires formal concept definitions. However, the DD supports separating different hierarchical views by providing explicit relationships between concepts, in contrast to nonspecific links as commonly provided by many terminologies.

Regarding the definition of context, considerable variability can be found in the literature, which is among other things due to the fact that many diverse disciplines are concerned with the broad notion of context. Hence, a detailed analysis of existing approaches to context is deliberately excluded from this thesis since it would have exceeded space limitations considerably. Regarding terminologies, it can be summarized from chapter 4 that although the need for context representation is recognized [Cimino, 1998], it is still in a rather experimental stage and is included in few terminologies only. If context is addressed, a precise definition is often lacking, though. For example, the context-sensitive information presentation provided by the Gießener Data Dictionary Server[57] is certainly useful, but is based on a rather intuitive understanding of context as “the clinical information that is being presented at the moment the information need is occurring” ([Ruan, Bürkle, et al., 2000], p. 719). Essentially, from the description of the algorithm given in [Ruan, Bürkle, et al., 2000], it seems that context (i.e. “the clinical information that is being presented”) is represented by a set of search terms for which the navigation algorithm is executed, returning information for the search terms as well as for terms that are in the data dictionary linked to the search terms. Note that such a search algorithm on the basis of relations between concepts could also be implemented on the basis of the DD model.

The issue of exploiting relationships between concepts is also common to many approaches to the representation of context. Specifically, a concept’s context could be defined as the set of concepts that are similar with regard to some measure of similarity, where the latter is based on the number of relations within a given set of concepts. This understanding of context defines a context as a subset of concepts and thus seems to be closer to our notion of view. This is an indication of the tight relation between the notions of context and view. Further, remember that one of the context properties is `C_Domain` which describes the set of domains that are of interest in the current context. Effectively, the extension of a domain is just the set of concepts which will be contained in a view that focuses on that domain (via `V_Domain`), which is equivalent to the above understanding of context. One important difference between the above notion of context and our notion of context is that while concepts are assigned manually to domains in the DD, the abovementioned mechanism, which employs relations for determining semantic similarity, allows for automatic classification of concepts into contexts. We have refrained from suggesting such an automatic approach since the development of an adequate measure of similarity is difficult and needs detailed examination[58]. Further, especially in the initial phase of content acquisition, the number of relations between concepts is expected to be insufficient in order to achieve reliable results.

[57] The navigation algorithm used by the GDDS is explained at the end of section 4.2. Briefly, the search for relevant information sources starts at a given concept and returns all information sources which are either attached directly to the search concept or to any of its related concepts.

[58] Examples of approaches are explained and examined in [Fellbaum, 1998].

6.4 What is „one“ Concept?

The greatest difficulty in developing the DD model consisted of delimiting the model elements from one another. Two important issues are discussed in the following, namely the question of what “one” concept is, and the related question of the delimitation of concepts and categories.

At first, there is the question of what “one” concept is, or in other words, how to determine whether two alternative definitions are describing the same “principal” meaning. The converse of this question is how to determine how many meanings an ambiguous term (in the sense of the semiotic triangle) has. As stated in section 2.1, although many disciplines are concerned with finding an answer to the question of what one unit of meaning is, no commonly accepted answer has been established yet. An example which illustrates this problem is the variability with which dictionaries describe the meanings of terms: when looking up the same term in different dictionaries, it is very likely that one will find a different number of meanings being explained.

In general, it must be concluded that this problem cannot be solved in the DD either. However, we argue that the definition of concept is less problematic in the DD because mainly specialist terminology will be defined in the DD rather than words of everyday language. Objectively determining the possible meanings of a word of everyday language is more difficult because all possible contexts in which the word may occur must be considered, and very subtle variations in meaning are possible, involving emotional issues. Such aspects are less relevant for specialist terminology because the context of specialist terms is never totally unrestricted but partially determined by the subject field (e.g. the domain of clinical trials).

A further argument for the feasibility of this approach is that despite the lack of agreement on the exact definition of terms, the principal meaning of terms is often generally accepted. A detailed example is given in the introduction, where e.g. experts agree that `relapse` means the reoccurrence of a disease after remission, but disagree on the exact definition with respect to which signs are to be diagnosed as a reoccurrence of the disease, etc. The generally accepted principal meaning is represented by means of the class `Concept`, and the different alternative – more or less precise – definitions are represented by the class `Definition`.

We expect the question of representing two definitions by means of ‘two concepts’ versus ‘one concept with two alternative definitions’ to arise especially in the cases of polysemous terms as well as definitions which exhibit a different context-specificity (cf. reason 5: Different Specificity of Domain Context). In both cases, experts should judge whether one or two (possibly related) concepts should be created. As a general guideline, very subtle distinctions should be represented rather as one concept with alternative definitions, in order to avoid an unmanageable number of unclearly delimited concepts.

As mentioned at the beginning of this section, a second issue, related to the question of what “one” concept is, regards the distinction between `Concept` and `Category`. This distinction is analogous to that between concepts and semantic types in the UMLS (cf. 4.1.4). With regard to the definition of concept in the semiotic triangle, it should be noted that categories (or semantic types, respectively) are essentially concepts as well. The delimitation of concepts from semantic types is an unresolved issue in UMLS [Burgun, Bodenreider, 2001]. There even occur some generic concepts (e.g. `<disease>`) which are represented in UMLS as both a concept and a semantic type. Note that this problematic distinction has motivated choosing the rather neutral name `belongsTo` for the association between `Concept` and `Category` instead of `instance-of` or `is-a`.

As an alternative to this strict distinction we had considered an approach in which categories can be defined as normal concepts as well and just play the additional role of a category for other concepts. On the one hand, this approach would be more flexible and would avoid some concepts having to be defined as categories as well, which is a form of redundancy. On the other hand, using concepts as categories entails a lot of critical problems. Most importantly, the main reason for deciding against this approach has been that it is unclear in which way alternative definitions of a category should affect the concepts assigned to that category. The initial purpose of categories as stated in requirement R2 is to provide a uniform template for the definition of similar concepts. With respect to this requirement it seems undesirable to have alternative definitions of categories. Further, certain decisions would have to be made which are critical with respect to extensibility towards building complex concepts. For example, cycles of the `belongsTo` relation can be permitted or forbidden, which is a well-known problem in meta-modeling[59] (cf. [Pan, Horrocks, 2001; Patel-Schneider, Fensel, 2002]).

6.5 Domain-Independence and Extensibility of the Model

One of the basic goals of this thesis is to design the DD model such that it is to a great extent domain-independent, flexible and extensible in order to make its application possible in other subject fields than that of clinical trials.

This aim is accomplished by having avoided the definition of any domain-specific model elements on the UML class level. In particular, domain-specific aspects are shifted to the instance level by defining the domain-independent classes `Category` and `Relationship`.

Alternatively, the relationships which can be used to link concepts could have been predefined through UML associations at the class `Concept` and, similarly, the types of concepts which are represented as categories could have been predefined as UML classes. In contrast to this, the advantage of our approach is that it allows for the introduction of arbitrary domain-specific relationships and categories together with templates which determine the structure of concept definitions without changing the DD model itself.

A consequence of this flexibility and domain-independence is that there are two levels of content in the DD, where each level is expected to be edited by different types of users. Before the DD is applied for collecting terms and their definitions in a certain domain, it needs to be adapted to the domain by predefining the categories and relationships which are useful in that domain. This task should be accomplished by users who are experienced in working with the DD. Afterwards, domain experts can enter definitions by filling appropriate category templates and establishing relations between concepts. Of course, this two-step process is not strictly sequential; it is possible to define new categories and relationships at any time as required.

[59] In combination with the definition of complex concepts in terms of a logical language, this touches on the famous paradox of Russell. Adapted to the DD model classes, the paradox would arise if categories could `belongTo` themselves and a language would allow for the expression of the category of all those concepts which do not `belongTo` themselves. That category would `belongTo` itself if and only if it did not `belongTo` itself.

7 Prototypical Implementation

7.1 Architectural Design: DD-Server and Onto-Builder

On the basis of the functional requirements identified in section 3.3 and a comparison with existing architectures of medical data dictionaries (cf. section 4.2), we propose the separation of the functionality into two applications: the **DD-Server** and the **Onto-Builder**.

The first application, the DD-Server, makes the contents of the DD itself (i.e. the repository of terms, concepts, definitions, etc.) available to other applications by providing an API for querying and editing the DD (as required in R17). The Onto-Builder is an application with a graphical user interface that provides the functionality for browsing and managing the content of the DD collaboratively and in a distributed manner (as required in R18 and R19). The Onto-Builder will also implement the functionality for the harmonization process as well as the quality assurance workflow (R16). Figure 7.1 below illustrates this architecture and also shows the integration of external terminologies (cf. R20).

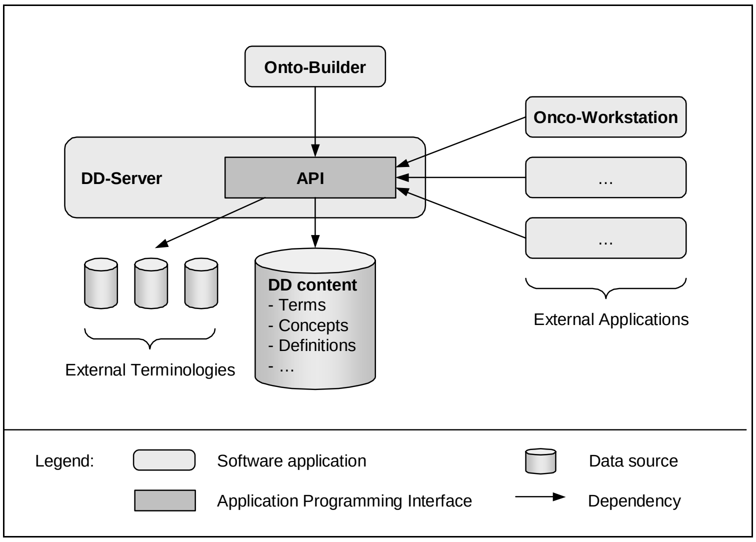


Figure 7.1: Architectural design illustrating the distinction between the DD-Server and the Onto-Builder.

Note that this strict distinction between the DD-Server and the Onto-Builder is not made in the existing Onto-Builder Version 1. As described in section 1.4, the existing first version of the DD is implemented as a relational database and does not provide an API for querying it. Of course, it is possible to retrieve terms and definitions from the DD via SQL[60] queries, but this requires the programmer of an external application to understand the details of the DD model. An API as proposed in this thesis for the DD provides a higher level of abstraction from the details of the model structure. An additional benefit of this higher level of abstraction is that

[60] SQL = Structured Query Language. ANSI standard language for accessing and manipulating relational databases.

changes to the structure of the DD model do not necessarily affect the API and thus do not require changes in the querying application.

7.2 Prototypical Implementation in Protégé

This section describes the prototypical implementation of the DD model developed in this thesis. The major purpose of this implementation is to examine the feasibility and possible benefits of implementing the DD on the basis of Protégé, a free and extensible software tool from Stanford University (USA) [Protégé, 2004].

7.2.1 Rationale for Choosing Protégé

In principle, there are two possible choices for implementation. Firstly, the DD model could be implemented in a relational database and the DD-Server as well as the Onto-Builder could be implemented in any programming language. The second possible choice is to reuse existing software components and tools as the basis for implementation. The benefit of the second choice compared to the first is obvious: given that components exist which closely fit the needs of the envisioned implementation, effort and costs can be reduced significantly.

Members of the DD project group therefore investigated opportunities for the second choice. This work was carried out in parallel to this thesis and concentrated on ontology tools since they provide functionalities for defining concepts. The tools were evaluated with regard to their suitability for implementing the DD model itself as well as the DD-Server and the Onto-Builder.

A review of existing tools revealed that many existing ontology tools are not sufficiently mature to be employed in a working environment because they are still in an experimental development stage and are thus mostly unstable and / or have little documentation and support [Hentschel, 2003]. Furthermore and most importantly, the requirement of being extensible in order to allow for the implementation of additional functionalities required for the DD is not fulfilled by most tools.

One of the few tools which were found to be sufficiently stable and mature to be suitable for implementing the DD, is Protégé.[61] Among the distinguishing features of Protégé are its extensibility (details are described in the following section 7.2.2) as well as its maturity while still being up-to-date. The development of Protégé started at the beginning of the eighties and since then has evolved significantly – including several complete reimplementations of the system – in order to adapt it to changing technologies, modeling methodologies and user requirements [Grosso, Eriksson, et al., 1999]. Thus, Protégé is a mature and well-tested software while still being under active development (e.g. implementation of multiuser capabilities).

The purpose of the implementational work carried out within the scope of this thesis is to demonstrate the feasibility and benefits of implementing the DD model as well as the functionality required for the DD-Server and the Onto-Builder on the basis of Protégé.

[61] Protégé is freely available under an open-source license from Stanford Medical Informatics [Protégé, 2004]. The current version of Protégé is Protégé 2.0 (as of April 2004).

7.2.2 The Protégé Model

As described in [Musen, Fergerson, et al., 2000] (p. 1), Protégé is a flexible "meta-tool" for the construction of knowledge acquisition tools in the sense that Protégé is a highly configurable tool which can be used to develop knowledge acquisition (or data entry) tools. How this goal is achieved in Protégé is explained in the following according to [Noy, Fergerson, et al., 2000].

The Protégé model is frame-based[62] and distinguishes between *classes* and *instances*, where classes are used to represent domain-relevant concepts, and instances to represent objects. Classes can be organized in a polyhierarchy using the is-a relation, allowing for multiple inheritance. The root of this hierarchy is the system class `:THING`[63]. Multiple classification is not allowed in Protégé, i.e. each instance belongs to exactly one class which is called its *type*[64].

The properties of classes are described by *template slots*. A template slot itself has properties (called *facets*), like the name of the slot, the allowed value type (e.g. integer, string, or instances of some class), as well as the cardinality (i.e. how many values are allowed to fill this slot). Template slots are propagated to the instances of the class where they become *own slots* and are filled with concrete values. Template slots are inherited to subclasses in the abovementioned inheritance hierarchy. Template slots are thus comparable to attributes in UML.

Figure 7.2 contains a screenshot of Protégé, showing the example project which is included in Protégé as a tutorial – a project describing concepts related to the newspaper domain. The purpose of this screenshot is to illustrate the basic features of Protégé described above.

Note that slots are so-called "first-class entities", i.e. they are defined independently of classes and can then be attached to multiple classes as their template slots. A common example is a slot `name` which is defined as a slot of value type string and can be reused as a template slot at multiple classes. When a slot is attached as a template slot to a specific class, the facets of the slot can be *overridden* at that class in order to be more restrictive, e.g. the allowed value type can be changed to a more specific type. Such facet overrides are inherited to subclasses, where they can be overridden again if desired.[65]

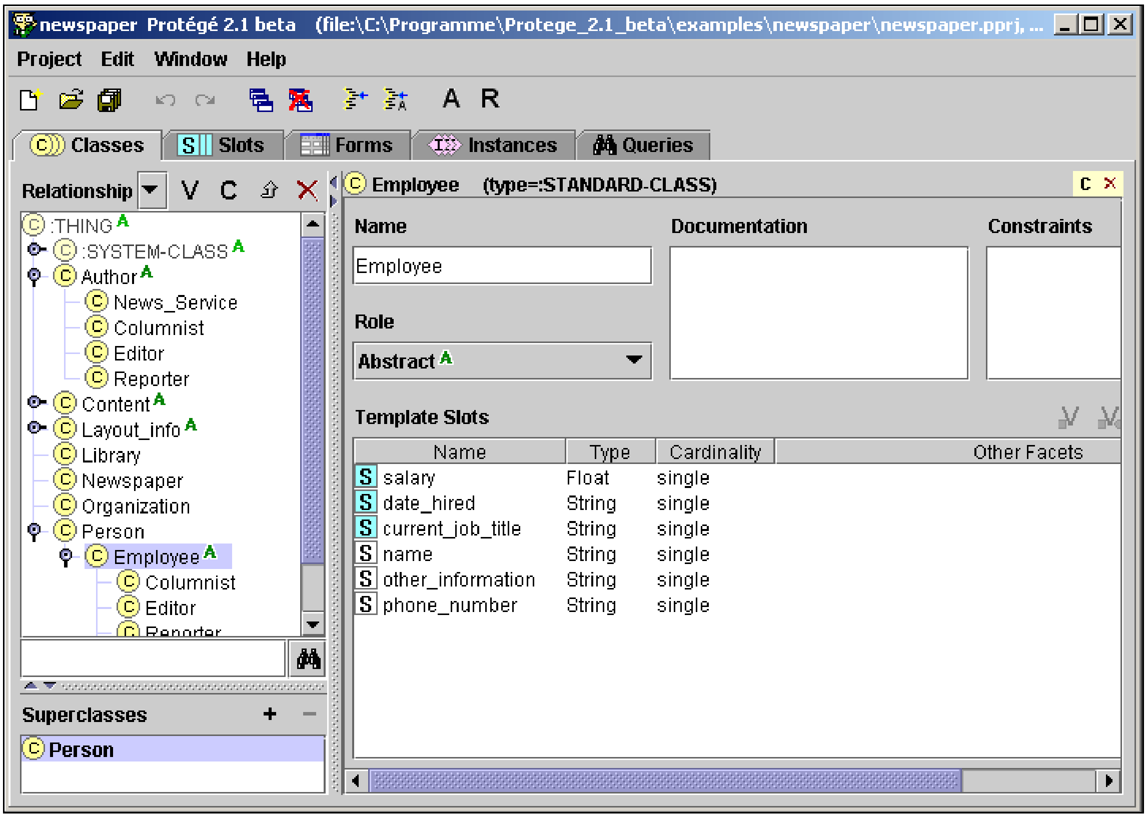
A feature which is very important with respect to Protégé's flexibility is its metaclass architecture. A *metaclass* is a class whose instances are classes themselves. In order to distinguish classes having "simple instances" (i.e. instances which are not classes) from metaclasses, the former are called *simple classes*. The notion of class comprises both metaclasses and simple classes. Arbitrary levels of metaclasses are possible, i.e. instances of a metaclass can be metaclasses again. As described above, each instance has exactly one direct type, which is no different for classes, i.e. each class has one direct type – its metaclass. The default metaclass for all user-defined classes is the Protégé built-in metaclass `:STANDARD-CLASS`.

[62] For an introduction to frame-based systems the reader is referred to [Luger, 2001].

[63] Names of system frames (i.e. built-in in Protégé in order to implement the Protégé model itself) are always capitalized and start with a colon in order to distinguish them from user-defined frames. Further examples of system frames are `:CLASS`, `:SLOT`, etc.

[64] More precisely, this is the *direct* type. An instance can have multiple *indirect types*, which include the superclasses of its direct type.

[65] The usage of facet overrides becomes clearer in the following section 7.2.3 where they are used for implementing the DD model.



Name: Employee

Role: Abstract

	Name	Type	Cardinality	Other Facets
S	salary	Float	single	
S	date_hired	String	single	
S	current_job_title	String	single	
S	name	String	single	
S	other_information	String	single	
S	phone_number	String	single	

Figure 7.2: Screenshot of the Protégé application with the newspaper example from the Protégé tutorial opened. The class hierarchy is visible on the left, the template slots of the class `Employee` are shown in the lower right part of the screen. Blue "S"-symbols indicate template slots which are attached directly to the class, whereas white "S"-symbols indicate slots which are obtained by inheritance from the superclass(es).

Note that the class hierarchy does not restrict which metaclasses are allowed as types of the classes. Although by default, each newly created class gets assigned the same type as its superclass, the type of a class can be changed to any other metaclass.

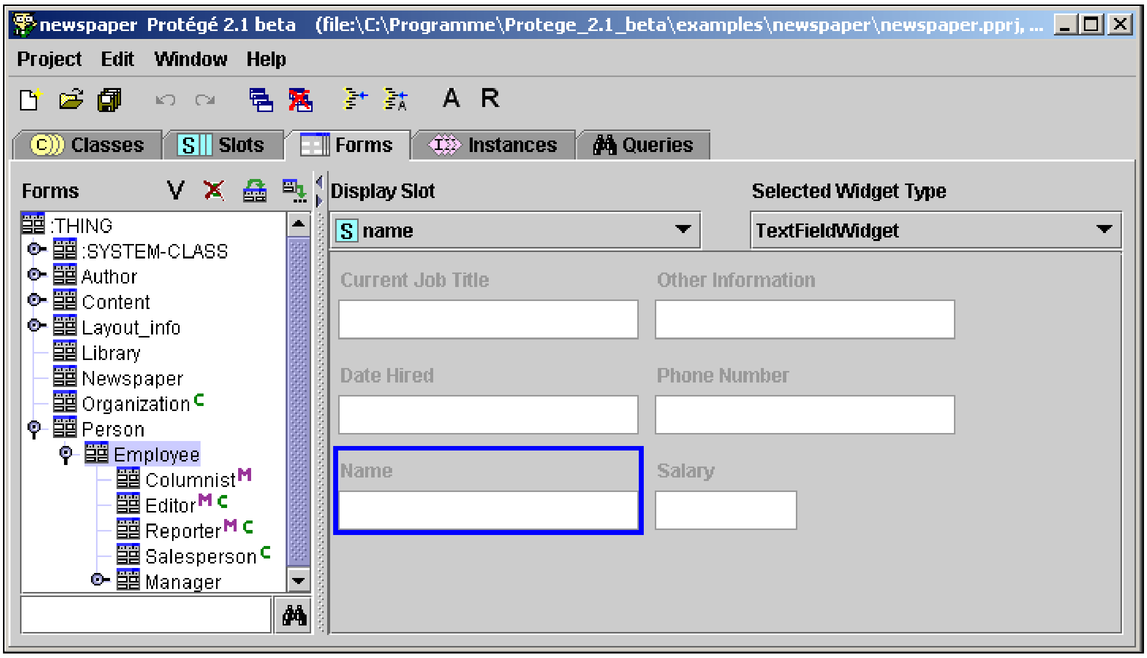
The template and own slot mechanism described above works analogously for metaclasses and the classes which are their instances. As a consequence, a class `C` can have template slots *and* own slots, where the latter are propagated to `C` from its metaclass and the former are either attached directly to the class or inherited from its superclass(es).[66] Examples of own slots of a class are the slots `:NAME` and `:DIRECT-SUPERCLASSES` (the latter contains as values references to the superclasses). Note that template and own slots behave differently with regard to inheritance. Only template slots are inherited – own slots are completely specific to a class and are neither inherited to subclasses nor propagated to instances of the class.

After having explained the model of Protégé, the meaning of the statement from the beginning of this section regarding Protégé as a meta-tool for the construction of knowledge acquisition

[66] Note that this is different from simple instances which can have own slots only. Attaching template slots to simple instances does not make sense since there are no instances of simple instances which could fill the corresponding own slots.

tools can be explained. The basis of Protégé as a meta-tool is its *form mechanism*. For each class that is defined, Protégé automatically generates a form for acquiring instances of that class. The form contains for each template slot of the class a so-called *widget* which is a user interface component for entering the value of the corresponding own slot at an instance. Protégé provides several widget types and automatically determines which widget type is most appropriate for a slot. Concerning the example of the class `Employee` given in figure 7.2 above, the default form contains six widgets – one per template slot, where, for example, the slot `name` (of type string) is acquired using the "TextFieldWidget". The slot `Salary` is acquired instead using the "FloatFieldWidget" which allows entry of float values only. If an invalid value is entered, this is indicated by a red frame which is displayed around the widget.

The form layout which is generated automatically by Protégé is highly customizable. The user can influence, for example, the size, order and layout of the widgets as well as select which particular widget type is chosen for acquiring an own slot value. Figure 7.3 shows the form of the class `Employee` being customized in the "Forms Tab".



newspaper Protégé 2.1 beta (file:\C:\Programme\Protege_2.1_beta\examples\newspaper\newspaper.pprj, ...

Display Slot: name

Selected Widget Type: TextFieldWidget

Figure 7.3: The "Forms Tab" shown in this screenshot allows for customization of the form for the selected class `Employee`. The currently selected widget (indicated by the blue frame) can be resized, moved around to another place on the form, or another widget type can be selected using the pull-down menu on the right. It is also possible to change the label of the widget (currently, it is "Name").

The form mechanism allows for the construction of a knowledge acquisition tool insofar as a knowledge engineer can define the classes with its template slots and customize the forms as desired. The resulting set of forms represents a knowledge acquisition tool which can be used by domain experts for acquiring instances by filling in these forms.

In case the widget types provided by Protégé are not sufficient for the needs of some particular project, it is possible to implement new widget types which then can be selected in the form customization process. The possibilities of extending Protégé are not limited to implementing new widgets, however, since a major design goal of Protégé has been to allow users to add their own components (called *plugins*) that implement additional functionality as required. Protégé maintains an extensive library of plugins on their website which contain many plugins contributed from users, e.g. plugins for visualization (e.g. OntoViz), for formulating rules and constraints on classes (e.g. AlgernonTab, JessTab), or for connecting to other resources (e.g. the UMLSTab allows the UMLS to be searched from within Protégé). Concerning implementation it should be mentioned that Protégé is implemented in Java, and is thus platform-independent.

7.2.3 Implementation of the DD Model in Protégé

As explained in the previous section, due to its flexible form mechanism, Protégé can be regarded as a meta-tool allowing for the creation of knowledge acquisition – or more general – data entry tools. This aspect makes Protégé suitable as the basis for implementing the Onto-Builder because the purpose of the Onto-Builder is to enter and edit content of the DD. The Onto-Builder can thus be considered as a data entry tool for the DD model, although the Onto-Builder will of course also implement more complex functionalities which go beyond a simple data entry tool, e.g. support of the harmonization process. Implementing these additional complex functionalities should be unproblematic, since the flexible plugin mechanism of Protégé allows for functional extensions.

In analogy to the general methodology of Protégé, the straightforward solution for implementing the Onto-Builder in Protégé is to represent the UML classes of the DD model as Protégé classes. The set of customized forms then assembles the Onto-Builder since the forms allow concepts, terms, definitions, etc. to be entered.

Although this straightforward solution is feasible, it is not optimal since it does not exploit similarities that exist between the DD model and the Protégé model. In particular, as explained in detail below, the DD model contains some classes that model behavior similar to the form mechanism in Protégé. The major aim of the implementational work carried out within this thesis is therefore to find an approach to implementing the DD model in Protégé that best exploits the Protégé form mechanism.

Let us briefly recapitulate those DD model classes which are important with regard to the abovementioned similarity to Protégé. As described in the section on semiformal definitions of concepts (5.4.2), a `Category` defines a `Template` consisting of `TemplateComponent`s (three subtypes of components are defined: `RelationalTemplComp`, `AttributiveTemplComp`, `TextualTemplComp`). Each `Concept` can be assigned to multiple categories with the effect that the templates of these categories determine the structure of each `SemiformalDefVersion` of that concept. In detail, a `SemiformalDefVersion` consists of a number of `FilledTemplate`s where each of these corresponds to – i.e. `fills` – exactly one `Template`. Analogously to the structure of `Template` and `TemplateComponent`s, a `FilledTemplate` consists of `FilledComponents` which contain the content of the `SemiformalDefVersion` entered into the corresponding `TemplateComponent`.

Implementing these DD model classes using the straightforward approach mentioned above would result in the following structure. A concrete template defined for some category would be a simple instance of the Protégé class `Template`. This template instance would – via own slots – maintain references to instances of the Protégé class `TemplateComponent` representing the sections of the template. Likewise, a concrete filled template would be an instance of

FilledTemplate, also maintaining references to instances of FilledComponent, respectively. The fills association that exists in the DD model between the classes Template and FilledTemplate as well as between TemplateComponent and FilledComponent would also be implemented in Protégé using template slots attached to FilledComponent and FilledTemplate, so that instances of these classes would have an own slot pointing to the particular template (or template component, respectively) which is being filled.

The drawback of this approach is that although the information about which templates are filled by a semiformal definition is represented correctly, the forms generated automatically by Protégé to acquire a semiformal definition will not ensure that the content entered actually conforms to the format defined by the corresponding templates. Consider for example the classes AttributiveTemplComp and AttributiveFilledComp. An AttributiveFilledComp has the attribute content, and the format of the value of this attribute is constrained by a corresponding instance of AttributiveTemplComp which specifies the allowedValueType as well as the minimum and maximum multiplicity of the value. Precisely this constraint would not be enforced by the forms generated automatically by Protégé. The generated form for AttributiveFilledComp would just contain a widget for filling the slot content, but the value entered would not be checked for conformance with the specified value type and multiplicity.

Being aware of this issue, the question is how the DD model can be implemented in order to achieve automatic generation of forms which, as far as possible, directly enforce the constraints specified by a specific category template when entering a semiformal definition. The remainder of this section describes the mapping of the DD model to the Protégé model by which this goal of exploiting the Protégé form mechanism is achieved. Tables 7.1, 7.2 and 7.3 given at the end of this section summarizes the mapping.

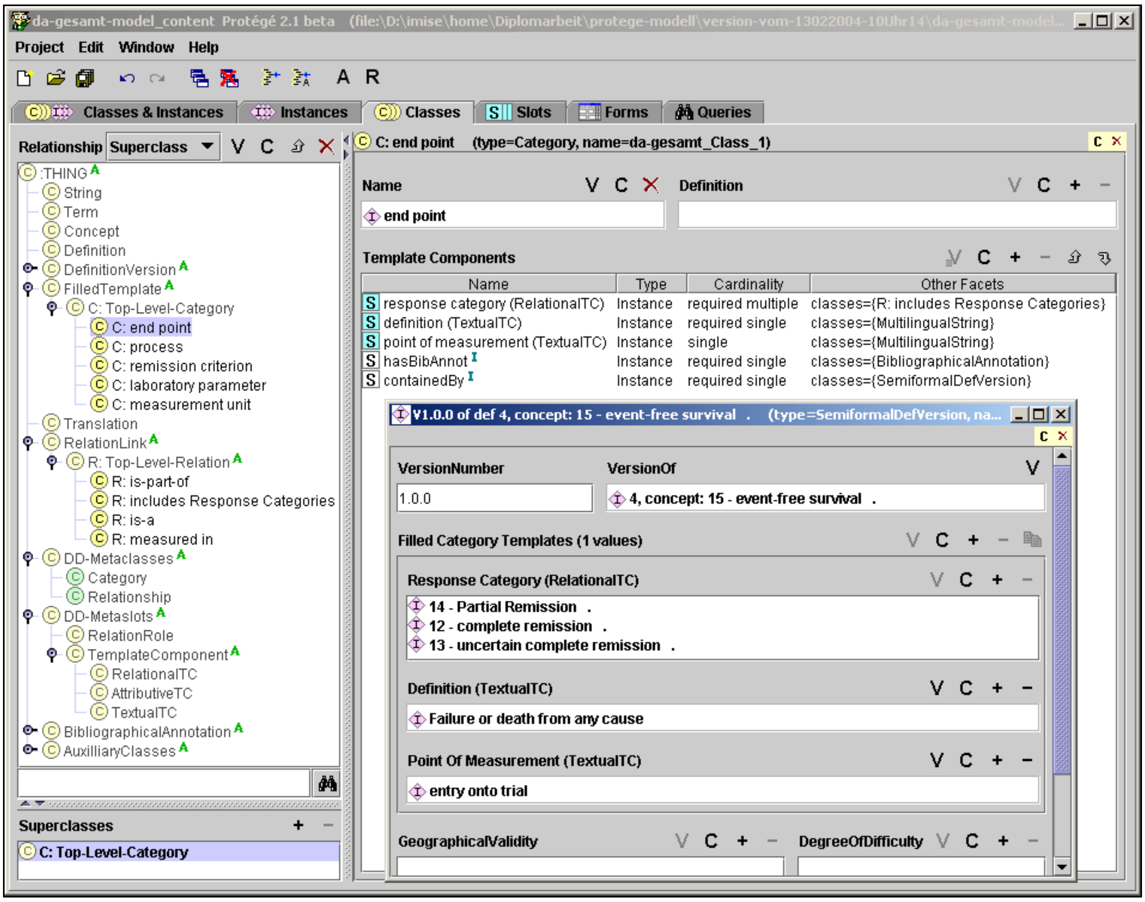
The fundamental observation which guides the implementation of the DD model in Protégé is that the DD template mechanism (i.e. the duality of templates and filled templates) is very similar to the form mechanism in Protégé. For example, instances of the DD class TemplateComponent correspond to template slots in Protégé because both define a template structure constraining the format of the content allowed to be entered, and analogously, FilledComponents correspond to own slots because both contain the content (or value) entered for specific instances.

In order to exploit Protégé's form mechanism, the DD class Category is modeled as a metaclass in Protégé. Consequently, concrete categories[67] are Protégé classes (instead of simple instances as in the straightforward approach described above) and hence can have template slots. As discovered in the previous paragraph, instances oft the DD class TemplateComponent correspond to template slots – we therefore implement this class as well

[67] The terminology needed for describing the implementation is somewhat complicated. This is due to the fact that the class and instance layer of the DD model are completely separated in UML but get mixed in the Protégé implementation because some UML classes are mapped to Protégé metaclasses and thus UML instances are mapped to Protégé classes. We try to avoid confusion by using the fixed-width font when a class name refers to the actual UML class (or the respective Protégé metaclass) and use a formulation like "a concrete category" in order to indicate that a UML instance (or Protégé class, respectively) of Category is meant.

as its three subclasses as *metaslots*. Metaslots are, in analogy to metaclasses, frames whose instances are slots – consequently, concrete template components are implemented as template slots. The metaclass `Category` is defined such that classes of type category can only have template slots of type `TemplateComponent`.

To summarize, a concrete category is defined in Protégé as a class of type `Category`. The template of that category is created by attaching template slots to the class. Due to the definition of the metaclass `Category`, only template slots of type `TemplateComponent` are allowed (see table 7.1 for details on how this is ensured). This construction leads to the result that the form generated automatically by Protégé for the newly created category contains one widget per template component and thus constitutes the template defined by the category. As described in the previous section, the layout of the form can be customized. This implementation approach therefore achieves the goal of exploiting the Protégé form mechanism for the template mechanism of semiformal definitions in the DD.



Name: end point

Name	Type	Cardinality	Other Facets
S response category (RelationalTC)	Instance	required multiple	classes={R: includes Response Categories}
S definition (TextualTC)	Instance	required single	classes={MultilingualString}
S point of measurement (TextualTC)	Instance	single	classes={MultilingualString}
S hasBibAnnot	Instance	required single	classes={BibliographicalAnnotation}
S containedBy	Instance	required single	classes={SemiformalDefVersion}

VersionNumber: 1.0.0

VersionOf: 4, concept: 15 - event-free survival .

Response Category (RelationalTC):
14 - Partial Remission .
12 - complete remission .
13 - uncertain complete remission .

Definition (TextualTC): Failure or death from any cause

Point Of Measurement (TextualTC): entry onto trial

Figure 7.4: Screenshot of the Onto-Builder prototype implemented in Protégé. The class hierarchy is shown on the left; yellow icons indicate simple classes and green icons indicate metaclasses.

Figure 7.4 shows an example of a category being defined. The example category shown is `end point`, which is used in section 3.1 for motivating the requirement R2 regarding the definition of categories and templates. This category defines three template components (shown with the blue “S”-symbol). The two bottom template slots (with the white “S”-symbol) are not template components but are slots unrelated to the template mechanism (they are inherited from the class `FilledTemplate`). The smaller window on the lower right shows a semiformal definition of the concept `event-free survival` using (i.e. filling) the template of the category `end point`.

As shown in the class hierarchy in figure 7.4, concrete categories are implemented as subclasses of `FilledTemplate`. Further, `FilledTemplate` is defined as an abstract class (indicated by the green "A" next to the class name). This is because instances of `FilledTemplate` are not direct instances of this class but rather are direct instances of the corresponding category. In other words, the association `fills` which exists in the DD model between `FilledTemplate` and `Template` is implemented using Protégé's instantiation mechanism. Therfore, the template slots of the category (= instances of `TemplateComponent`) become own slots for the filled template, which allows Protégé's form mechanism to be exploited.

The remaining DD classes are implemented using the straightforward approach, i.e. UML classes of the DD model are mapped to Protégé simple classes, UML instances to Protégé instances, UML attributes to Protégé template slots, and UML associations to a pair of inverse template slots. For example, `Concept` is a simple class which has a template slot `alternativeDefinitions`; concrete concepts are thus simple instances in Protégé with own slots pointing to instances of `Definition`. `Definition` has in turn a template slot with allowed value type "instances of `DefinitionVersion`". `DefinitionVersion` is (in complete analogy to the UML model) an abstract class with the two subclasses `SemiformalDefVersion` and `InformalDefVersion`; a semiformal definition points to instances of concrete categories, representing the filled templates which a semiformal definition consists of.

An overview of the major implementation aspects explained above is given in the tables that follow. The tables list only those classes which contribute significantly to achieving the broadest possible exploitation of Protégé's form mechanism.

Table 7.1 summarizes those classes which are relevant to the template mechanism (i.e. how categories influence the structure of semiformal definitions). The classes of the DD model are listed in the first column and the second column shows the Protégé constructs to which the UML classes and instances are mapped.

UML class in the DD model	Implementation of UML classes and instances in Protégé
Category	**UML class:** metaclass overridden :DIRECT-TEMPLATE-SLOTS[68] so that only slots of type TemplateComponent may be attached as template slots to classes of type Category **UML instance:** simple class
TemplateComponent and its subclasses RelationalTemplComp, AttributiveTemplComp, TextualTemplComp	**UML class:** metaslot overridden :DIRECT-DOMAIN so that template slots of this type may only be attached to classes of type Category overridden :SLOT-VALUE-TYPE of the three subclasses in order to ensure that the corresponding own slot (= filled components) will be of the correct value type, e.g. a multilingual string in the case of a textual template component **UML instance:** template slot
Template	**UML class:** – Template components are attached directly to the category, hence a separate class Template is unnecessary. **UML instance:** represented implicitly by the Protégé form which is generated automatically for a concrete category
FilledTemplate	**UML class:** simple class **UML instance:** simple instance that instantiates the category which is filled
FilledComponent and its subclasses RelationalFilledComp, AttributiveFilledComp, TextualFilledComp	**UML class:** – **UML instance:** own slot attached to instance of FilledTemplate

Table 7.1: Summary of the mapping of those classes relevant to implementing the DD template mechanism on the basis of Protégé's form mechanism.

[68] :DIRECT-TEMPLATE-SLOTS is a Protégé built-in template slot attached to every metaclass. We do not want to go into further technical details here, and the explanation given in the table regarding the purpose / effect of the override should suffice. The same holds for the other system slots which are mentioned in the table, e.g. :DIRECT-DOMAIN.

The following table 7.2 describes how the UML associations that exist between the classes listed in the previous table 7.1 are implemented in Protégé.

UML association in the DD model	Implementation of UML association in Protégé
`Template consistsOf TemplateComponent`	The attachment of template slots of type (metaslot) `TemplateComponent` to a concrete category implements the `consistsOf` association.
`FilledTemplate consistsOf FilledComponent`	Instances of `FilledComponent` are represented implicitly through the own slots at instances of `FilledTemplate`.
`FilledTemplate fills Template`	The filling of templates is represented through instantiation of the corresponding `Category` (the filled template is an instance of a category).
`FilledComponent fills TemplateComponent`	Analogously to `FilledTemplate`, the filling of template components is represented through instantiation of the corresponding `Category` (the filled template component is an own slot containing the value for a template slot – the template component).

Table 7.2: Summary of the mapping of the associations relevant to implementing the DD template mechanism on the basis of Protégé's form mechanism.

The last table given below shows the implementation of the classes concerning the representation of relations in the DD model (i.e. `Relationship`, `RelationalRole`, `RelationalLink`, `FilledRole`). These are implemented in analogy to categories, templates and filled templates and are therefore not discussed in detail but only shown in table 7.3.

UML class in the DD model	Implementation of UML classes and instances in Protégé
`Relationship`	**UML class:** metaclass overridden `:DIRECT-TEMPLATE-SLOTS` so that only slots of type `RelationalRole` may be attached as template slots to classes of type `Relationship` **UML instance:** simple class
`RelationalRole`	**UML class:** metaslot overridden `:DIRECT-DOMAIN` so that template slots of this type may only be attached to classes of type `Relationship` overridden `:SLOT-VALUE-TYPE` to ensure that the corresponding own slot (= instances of `FilledRole`) will be filled with a reference to a concept **UML instance:** template slot
`RelationalLink`	**UML class:** simple class **UML instance:** simple instance that instantiates the corresponding relationship
`FilledRole`	**UML class:** – **UML instance:** own slot attached to instance of `RelationalLink`

Table 7.3: Summary of the mapping of the class `Relationship` and related classes, which are also implemented on the basis of Protégé's form mechanism.

Thus far, we have described the process of implementing parts of the DD model in Protégé. This constitutes the major part of the implementational work. The second part of the prototypical implementation consists in the customization of the forms generated automatically by Protégé on the basis of the DD model classes and metaclasses. As indicated at the beginning of this section, this set of forms assembles the Onto-Builder. However, the automatically generated layout is not very intuitive, and therefore we have customized these forms extensively (e.g. changed the widget type, their label, and layout) in order to achieve a more usable and intuitive user interface. Of course, the current user interface of the Onto-Builder prototype (as shown in figure 7.4) is still provisional but is, as far as we have been able to determine, the optimum which can be achieved within the scope of existing Protégé widget types, plugins and the opportunities which they provide. Adapting and optimizing the user interface is unproblematic, however, due to the extensibility of Protégé by additional Java components.

At the current stage of implementation, the DD contents can be edited using the Onto-Builder prototype, but some constraints and mechanisms which should be handled automatically need to be provided for manually, e.g. unique identifiers of terms and concepts should be generated automatically but currently need to be entered by the user. Further, implementation of the context and view mechanism as well as additional functionalities (the harmonization process, etc.) is not addressed within this thesis.

7.2.4 Discussion of the Prototypical Implementation

As mentioned earlier, the purpose of the prototypical implementation is to demonstrate the feasibility as well as to examine the benefits of implementing the DD and the Onto-Builder on the basis of Protégé. Regarding feasibility, it can be concluded that the previous section has shown that Protégé's design as a meta-tool allows for the relatively quick implementation of a prototype because part of the user interface of the Onto-Builder is generated automatically through the Protégé forms.

In particular, using and extending Protégé's form mechanism is beneficial on two levels. Firstly, by implementing the DD model in Protégé, the automatically generated forms assemble a prototype of the Onto-Builder which allows for the entry of content, i.e. terms, concepts, definitions, etc. Secondly, by implementing some DD classes as Protégé metaclasses or metaslots, the form mechanism can also be used for generating the category templates for entering semiformal definitions.

In both cases, the effort and time needed for implementation is considerably less compared to an implementation "from scratch". Implementing the additional required functionality as well as improving the user interface should also be possible due to Protégé's plugin mechanism which allows for extensions.

The implementation in Protégé has further revealed some interesting effects on the DD model. For example, since instances of the DD class `Category` are implemented in Protégé as classes instead of simple instances, it is possible – without any additional modeling or programming effort – to define a polyhierarchy of categories, in which template components are inherited to subcategories. It is even possible to specialize inherited template components using Protégé's override mechanism. Such an inheritance hierarchy of categories can certainly be useful but requires further investigation in order to evaluate the concrete consequences with regard to semiformal concept definitions.

As a final note, we need to relate the prototypical implementation in Protégé to the architectural design proposed in section 7.1. At the current stage, the DD content is contained in a Protégé project which can be saved in different formats, e.g. text file, database, XML. The Onto-Builder is prototypically implemented by customizing Protégé. The prototypical implementation does not contain a separate DD-Server yet. It is suggested that the DD-Server be implemented as a Java-RMI[69] application that accesses the Protégé project via the Protégé API, since Protégé itself also provides an RMI server for remote and multi-user access to Protégé projects which could be extended. Details of the design and implementation, however, extend beyond the scope of the present work and remain to be developed.

[69] RMI = Remote Method Invocation. RMI allows clients to call methods on a remote computer system (see [Balzert, Helmut, 2000]). In the case of the DD, the clients are the applications that access the DD-Server, and the DD-Server is the remote computer system which provides methods for querying the DD for terms, definitions, etc.

8 Conclusion

8.1 Summary of Results

The present work has described the specification of a model for a data dictionary which allows for multiple alternative definitions of concepts as well as for context and view representation. Starting from the idea of a Data Dictionary for clinical trials as described in section 1.2 and an existing first version of the software tool Onto-Builder supporting the acquisition of terms and their definitions (introduced in section 1.4), a comprehensive requirements analysis was conducted in chapter 3.

This analysis focused on the contents which should be accommodated by the DD, initially neglecting the demand for alternative definitions. Following that, we examined the reasons for the occurrence of differences in definitions which emphasized the need for alternative definitions since differences may be justified and thus not reasonably harmonizable. For instance, differences may be necessary due to certain circumstances (such as different geographical locations of clinical trials) or if the same concept is described differently for different purposes (e.g. describing different details according to different points of view, or formulated in different languages). After having identified the reasons for differences in definitions, the requirements analysis continued, deriving requirements regarding the representation of alternative definitions, contexts and views, as well as the harmonization of alternative definitions. The analysis finished with collecting the broad functional requirements of the DD (e.g. with respect to editing DD content).

Existing terminology systems were reviewed in chapter 4 on the basis of these requirements. The main outcome here was that although some advancements towards the representation of different views (in the sense of consistent alternative definitions) as well as context-based information presentation can be observed, existing systems do not meet the requirement of inconsistent alternative definitions of a concept.

The main part of the thesis comprised the elaboration of the DD model described in chapter 5, which exhibits considerable improvements and extensions as compared to the existing first version. A first fundamental model choice is the adoption of a concept-centered approach according to the term-concept distinction defined by the semiotic triangle. This allows for the correct representation of synonymy (including terms in different languages) and ambiguity, and consequently for comparisons of alternative definitions of a concept regardless of which term is used to designate it.

Secondly, the concept-centered approach has been extended in order to accommodate alternative definitions of the “same” concept. This required a slight deviation from the common understanding of what “one concept” is, since from a strictly extensional point of view alternative definitions would actually always be understood as describing different concepts. The solution proposed here is to regard a concept as a means for grouping the alternative definitions which are to be compared during harmonization because they are judged to describe the same principal meaning.

Thirdly, the occurrence of differences in definitions identified in the requirements analysis have been classified into three groups, namely 1) content, 2) language, and 3) point of view. While the first refers to substantial differences in definitions (e.g. due to different expert opinions), the latter two groups refer to differences in definitions which are consistent with regard to content, i.e. the same substantial definition can be translated to different languages or formulated from different points of view. We have developed a view mechanism (summarized below) which provides support for representing only the first type of difference in definitions by means of alternative definitions, while the latter two issues are represented as different views on one alternative definition as far as possible[70].

Concisely, the above issues are resolved by means of four main levels in the DD model: terms (words which are to be defined), concepts (the principal meanings of terms, broadly covering synonymous and ambiguous terms), alternative definitions (inconsistent variants of definitions of the same concept), and consistent variants of the same alternative definitions (translations, and different points of view).

Regarding the format of definitions, the DD model allows for informal definitions (free text) as well as for semiformal definitions. The latter are based on templates which are defined for categories (reflecting groups of similar concepts), predetermining the structure of definitions of a concept according to the categories to which it belongs. While informal definitions are especially necessary to record existing definitions from various sources, the semiformal format is desirable against the background of the desired harmonization since it supports uniform definitions which also facilitate the analysis regarding the differences and similarities of alternative definitions.

Further, relations between concepts can be defined, contributing to precise definitions. According to the distinction of concepts (as the principal, uncontroversial meaning) and their alternative definitions, two kinds of relational links between concepts can be created: definition-independent and definition-dependent links. The latter are part of semiformal definitions, while the former are established directly between concepts and are thus valid for all their alternative definitions. The model allows for relations of arbitrary arity as well as for specifying multiplicity and category constraints (i.e. which category concepts must belong to in order to fill a relational role).

The support for harmonized definitions has also changed significantly as compared to the first version of the DD model. In the first version, it was only possible to create one "consensus suggestion" for a set of alternative definitions. The DD model developed within this thesis provides the possibility to indicate the scope of validity of definitions, instead, through which explicit marking of a definition as a consensus definition becomes unnecessary. This solution also allows for multiple harmonized definitions with different scopes of validity (if complete consensus is impossible).

Developing the context and view representation for the DD comprised two steps. First, general definitions were given and analyzed with regard to their relation. In summary, a view is said to be a projection of a base model containing only the content relevant in the current context, whereas context is understood in the sense of situative context comprising the characteristics of the current situation. The second step involved refining the general definitions towards concrete

[70] As discussed in section 6.2 this ideal solution is achieved only partially due to the complexity of the problem. For example, if the same substantive definition is formulated at different degrees of difficulty, this will be realized as alternative definitions which indicate the respective degree of difficulty.

definitions which are based on the DD model. Context is modeled in the DD by a set of context properties and content can be marked accordingly as being relevant in certain contexts. A view in the DD is a coherent subset of DD content which can be computed as a combination of several view axes. Each view axis selects content on the basis of a certain criterion (e.g. it may select concepts according to category or domain) and depends on context properties as its parameters (i.e. with respect to the previous example, the context properties specify which categories or domains are relevant in the current situation).

The elaboration of the DD model concludes with a discussion of the model developed (cf. chapter 6), including an evaluation against the requirements identified. Finally, a partial prototypical implementation in Protégé is presented in chapter 7 which focuses on evaluating the feasibility and benefits of implementing the DD model as well as the Onto-Builder (as the tool for browsing and managing DD content) in Protégé. It was shown that Protégé's flexible metaclass architecture can be exploited for implementing the template mechanism of the DD model.

8.2 Outlook for Further Developments

The immediate steps which should follow this work include, for one thing, the continuation of the implementation of the DD model as well as of the Onto-Builder. In so doing, the projects being conducted in parallel with respect to user modeling (access rights for contents and functions, user-tracking while editing contents) as well as the quality assurance cycle should be integrated. For another, the model and the proposed context and view mechanism should be evaluated by means of a comprehensive set of sample contents.

Moreover there are a range of expansion possibilities for the model which could be investigated. Among these are firstly the two requirements left open regarding the representation of rules as well as the integration of documents with additional relevant knowledge about concepts.

Apart from this, the representation of relations is a complex topic such that a lot of refinements of the current model are conceivable, for example the definition of different types of relations (hierarchical, transitive, symmetric, etc.) as well as relation hierarchies. A comprehensive analysis of issues regarding relations is given in [Loebe, 2003]. Future work could take this analysis further and examine which of the features described are of the greatest relevance to the DD and how they could be integrated.

Furthermore, the semantics of commonly-used relations, e.g. the `is-a` or the `part-of` relation, is a current research topic in the field of ontology and is being addressed by the Onto-Med research group in their development of the top-level ontology GOL [Heller, Herre, 2003]. It is planned that this work will be integrated with the DD in order to achieve ontologically founded concept definitions. The natural language definitions contained in the DD are of benefit to the GOL project since they can be analyzed in order to discover additional requirements for GOL.

Moreover, the understanding of concept as the “principal” meaning which is uncontroversial and common to all alternative definitions of a concept could be developed further. At the current stage of the DD model, a concept is described by a set of terms, a set of categories the concept belongs to, (definition independent) relational links to other concepts, a set of domains in which it is considered relevant, and – of course – its alternative definitions. Future work could investigate the possibilities for making the principal meaning more explicit, e.g. by definition components directly at the concept which may be refined and extended consistently by alternative definitions. The kinds of restrictions which are required here need to be determined.

Related to this issue is the probably the most exciting and, at the same time, most difficult question of how concept hierarchies, inheritance, and complex concepts can be provided in combination with alternative definitions. Examining the extent to which existing solutions are applicable in combination with alternative definitions of concepts is a research project in its own right. For this reason, the present work has deliberately excluded these issues.

References

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.) 2003. *The Description Logic Handbook. Theory, Implementation and Applications.* Cambridge (UK): Cambridge University Press.

Balzert, H. 1999. *Lehrbuch der Objektmodellierung: Analyse und Entwurf.* Lehrbücher der Informatik. Berlin: Spektrum Akademischer Verlag.

Balzert, H. 2000. *Lehrbuch der Software-Technik.* Vol. 1. Berlin: Spektrum Akademischer Verlag.

Bodenreider, O. 2001. Medical Ontology Research: A Report to the Board of Scientific Counselors of the Lister Hill National Center for Biomedical Communications. May 17. Bethesda (Maryland): U.S. National Library of Medicine.

Booch, G., Jacobson, I., Rumbaugh, J. 1999. *The Unified Modeling Language User Guide.* The Addison-Wesley Object Technology Series. Reading (Massachusetts): Addison-Wesley.

Bouquet, P., Serafini, L., Brézillon, P., Benerecetti, M., Castellani, F. (eds.) 1999. *Modeling and Using Context.* Proceedings of the Second International and Interdisciplinary Conference, CONTEXT'99, Sept 9-11, Trento, Italy. Lecture Notes in Artificial Intelligence, Vol. 1688. Berlin: Springer.

Brachman, R. J. 1983. What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer* 16(10)**:**30-36.

Burgun, A., Bodenreider, O. 2001. Aspects of the Taxonomic Relation in the Biomedical Domain. In: Welty, C., Smith, B. (eds.) *Formal Ontology in Information Systems: Collected Papers from the Second International Conference.* Oct 17-19, Ogunquit, Maine, USA. p. 222-233. ACM Press.

Bürkle, T. 2000. Klassifikation, Konzeption und Anwendung medizinischer Data Dictionaries [Habilitation thesis]. Justus Liebig University Gießen, Faculty of Medicine.

Cheson, B. D., Horning, S., Coiffier, B., Shipp, M. A., Fisher, R. I., et al. 1999. Report of an International Workshop to Standardize Response Criteria for Non-Hodgkin's Lymphomas. *Journal of Clinical Oncology* 17(4)**:**1244-1253.

Cimino, J. J. 1998. Desiderata for Controlled Medical Vocabularies in the Twenty-First Century. *Methods of Information in Medicine* 37(4-5)**:**394-403.

Cimino, J. J., James, J. 2000. From Data to Knowledge through Concept-oriented Terminologies: Experience with the Medical Entities Dictionary. *Journal of the American Medical Informatics Association* 7(3)**:**288-297.

de Keizer, N. F., Abu-Hanna, A., Zwetsloot-Schonk, J. H. M. 2000. Understanding Terminological Systems I: Terminology and Typology. *Methods of Information in Medicine* 39(1)**:**16-21.

Diehl, V., Sieber, M., Rüffer, U., Bredenfeld, H., Breuer, K., et al. 2001. Studienprotokolle der Primärtherapie: LPHD-Studie, HD10 für limitierte Stadien, HD11 für intermediäre Stadien, HD12 für fortgeschrittene Stadien. Cologne: German Hodgkin Lymphoma Study Group.

DIN 2342-1 1992. *Begriffe der Terminologielehre*. Berlin: Beuth.

Dolin, R. H., Spackman, K., Abilla, A., Correia, C., Goldberg, B., et al. 2001. The SNOMED RT Procedure Model. In: Bakken, S. (ed.) *Visions of the Future and Lessons from the Past: Proceedings of the 2001 AMIA Annual Symposium*. Nov 3-7, Washington DC, USA. p. 139-143. Bethesda (Maryland): American Medical Informatics Association.

Fellbaum, C. (ed.) 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. Cambridge (Massachusetts): MIT Press.

Grosso, W. E., Eriksson, H., Fergerson, R. W., Gennari, J. H., Tu, S. W., et al. 1999. Knowledge Modeling at the Millennium: The Design and Evolution of Protégé-2000 [Internet]. In: Anonymous (ed.) *Proceedings of the Twelfth Knowledge Acquisition for Knowledge-Based Systems Workshop*. Oct 16-21, Banff, Alberta, Canada. Cited 23.01.2004. Available from: http://sern.ucalgary.ca/KSI/KAW/KAW99/papers/Grosso1/kmatm.pdf

Heller, B. 1999. Konzeptvorschlag Data Dictionary. Discussion Proposal for Telematics Workshop.

Heller, B., Herre, H. 2003. Formal Ontology and Principles of GOL. Onto-Med Report No. 1. Research Group Ontologies in Medicine, University of Leipzig.

Heller, B., Lippoldt, K., Kühn, K. 2003a. Guideline for Creating Medical Terms. Onto-Med Report No. 4. Research Group Ontologies in Medicine, University of Leipzig.

Heller, B., Lippoldt, K., Kühn, K. 2003b. Handbook Onto-Builder: Part I: Construction of Medical Terms. Onto-Med Report No. 5. Research Group Ontologies in Medicine, University of Leipzig.

Heller, B., Löffler, M. 2002. Telematik und rechnerbasiertes Qualitätsmanagement in einem Kommunikationsnetzwerk. *Antrag zur Förderung eines medizinischen Netzwerkes bei malignen Lymphomen. Förderperiode II*. German Federal Ministry for Education and Research (BMBF), Institute for Medical Informatics, Statistics und Epidemiology (IMISE), University of Leipzig.

Hentschel, S. 2003. Evaluation of Ontology Tools. Internal Report. Research Group Ontologies in Medicine, University of Leipzig.

Huff, S. M., Rocha, R. A., McDonald, C. J., De Moor, G. J. E., Fiers, T., et al. 1998. Development of the Logical Observation Identifier Names and Codes (LOINC) Vocabulary. *Journal of the American Medical Informatics Association* 5(3)**:**276-292.

ICH 1996. ICH Harmonised Tripartite Guideline: Guideline for Good Clinical Practice (GCP) E6. International Conference on Harmonisation of Technical Requirements for Registration of Pharmaceuticals for Human Use.

Lehmann, T., zu Bexten, E. M. (eds.) 2002. *Handbuch der Medizinischen Informatik*. Munich: Carl Hanser.

Leiner, F., Gaus, W., Haux, R., Knaup-Gregori, P. 1999. *Medizinische Dokumentation. Lehrbuch und Leitfaden für die Praxis*. 3. ed. Stuttgart: Schattauer.

Loebe, F. 2003. An Analysis of Roles: Towards Ontology-Based Modelling [Master's Thesis]. Onto-Med Report No. 6. Research Group Ontologies in Medicine, University of Leipzig.

LOINC Committee 2004a. Logical Observation Identifier Names and Codes (LOINC®) Users' Guide [Internet]. February 2004. Indianapolis: Regenstrief Institute. Cited 08.04.2004. Available from: http://www.loinc.org/download/loinc/guide/LOINCManual.pdf

LOINC Committee 2004b. LOINC® Database Version 2.12 [Internet]. Released Feb 9, 2004. Indianapolis: Regenstrief Institute. Cited 10.04.2004. Available from: http://www.loinc.org

Luger, G. F. 2001. *Künstliche Intelligenz: Strategien zur Lösung komplexer Probleme*. 4. ed. Munich: Addison-Wesley, Pearson Studium.

Lyons, J. 1995. *Einführung in die moderne Linguistik*. 8. ed. C. H. Beck Studium. Munich: Beck.

McDonald, C. J., Huff, S. M., Suico, J. G., Hill, G., Leavelle, D., et al. 2003. LOINC, a Universal Standard for Identifying Laboratory Observations: a 5-year Update. *Clinical Chemistry* 49(4)**:**624-633.

Musen, M. A., Fergerson, R. W., Grosso, W. E., Noy, N. F., Crubézy, M., et al. 2000. Component-Based Support For Building Knowledge-Acquisition Systems. Report No. SMI-2000-0838. Stanford Medical Informatics, Stanford University (USA).

NLM 2003. UMLS® Knowledge Sources. 14. ed. (November Release, 2003AC). Bethesda (Maryland): U.S. National Library of Medicine. Cited 10.02.2004. Available from: http://www.nlm.nih.gov/research/umls/documentation.html

Noy, N. F., Fergerson, R. W., Musen, M. A. 2000. The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility. In: Dieng, R., Corby, O. (eds.) *Knowledge Acquisition, Modeling and Management: 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'2000)*. Oct 2-6, Juan-les-Pins, France. p. 17-32. Lecture Notes in Computer Science, Vol. 1937. Berlin: Springer.

Ogden, C. K., Richards, I. A. 1923. *The Meaning of Meaning*. New York: Harcourt, Brace & Co.

OMG 2003a. OMG Unified Modeling Language Specification (v1.5). Released 01.03.2003. Object Management Group. Cited 02.04.2004. Available from: http://www.omg.org/docs/formal/03-03-01.pdf

OMG 2003b. UML 2.0 OCL Specification. OMG Final Adopted Specification. Released 14.10.2003. Object Management Group. Cited 02.04.2004. Available from: http://www.omg.org/docs/ptc/03-10-14.pdf

OpenGALEN 2004a. GALEN Frequently Asked Questions [Internet]. Cited 15.04.2004. Available from: http://www.opengalen.org/technology/galen-faq.html

OpenGALEN 2004b. OpenGALEN: Making the impossible very difficult [homepage]. Cited 15.04.2004. Available from: http://www.opengalen.org

Pan, J. Z., Horrocks, I. 2001. Metamodeling Architecture of Web Ontology Languages [Internet]. In: Anonymous (ed.) *Proceedings of the First Semantic Web Working Symposium (SWWS'01)*. Jul 30 - Aug 1, Stanford, California, USA. p. 131-149. Cited 17.12.2003. Available from:
http://www.semanticweb.org/SWWS/program/full/SWWSProceedings.pdf

Patel-Schneider, P. F., Fensel, D. 2002. Layering the Semantic Web: Problems and Directions. In: Horrocks, I., Hendler, J. A. (eds.) *The Semantic Web - ISWC 2002: Proceedings of International Semantic Web Conference*. Jun 9-12, Chia, Sardinia, Italy. p. 16-29. Lecture Notes in Computer Science, Vol. 2342. Berlin: Springer.

Pfreundschuh, M., Trümper, L., Klöss, M., Löffler, M. 2001. Randomised Study Comparing 6 and 8 Cycles of Chemotherapy with CHOP (Cyclophosphamide, Doxorubicin, Vincristine and Prednisone) at 14-day Intervals (CHOP-14), both with or without the Monoclonal anti-CD20 Antibody Rituximab in Patients aged 61 to 80 Years with Aggressive Non-Hodgkin's Lymphoma (Including Amendment). Short Title: RICOVER-60. Study Number: DSHNHL 1999-1A. Homburg: German High-Grade Non-Hodgkin's Lymphoma Study Group.

Pocock, S. J. 1983. *Clinical Trials. A Practical Approach*. Chichester: John Wiley & Sons.

Protégé 2004. The Protégé Ontology Editor and Knowledge Acquisition System [homepage]. Stanford Medical Informatics, Stanford University. Cited 22.04.2004. Available from: http://protege.stanford.edu

Pschyrembel, W., Dornblüth, O. (eds.) 1998. *Pschyrembel Klinisches Wörterbuch*. 258. ed. Berlin: Walter de Gruyter.

Rector, A. L. 1999. Clinical Terminology: Why Is it so Hard? *Methods of Information in Medicine* 38(4-5)**:**239-252.

Rector, A. L., Nowlan, W. A., Glowinski, A. 1993. Goals for Concept Representation in the GALEN Project. In: Safran, C. (ed.) *Proceedings of the 17th Symposium on Computer Applications in Medical Care (SCAMC)*. p. 414-418. Washington: McGraw-Hill.

Rector, A. L., Rogers, J. 1999. Ontological Issues in using a Description Logic to Represent Medical Concepts: Experience from GALEN [Internet]. *Terminology and Natural Language in Medicine: Proceedings of the IMIA WG6 Workshop*. November, Phoenix, Arizona. Cited 23.11.2003. Available from:
http://www.opengalen.org/technology/papers.htm

Rector, A. L., Solomon, W. D., Nowlan, W. A., Rush, T. W., Zanstra, P. E., et al. 1995. A Terminology Server for Medical Language and Medical Information Systems. *Methods of Information in Medicine* 34**:**147-157.

Rogers, J., Roberts, A., Solomon, D., van der Haring, E., Wroe, C., et al. 2001. GALEN Ten Years On: Tasks and Supporting Tools. In: Patel, V., Rogers, R., Haux, R. (eds.) *MedInfo 2001*. p. 256-260. Studies in Health Technology and Informatics, Vol. 84. Amsterdam: IOS Press.

Rogers, J. E., Rector, A. L. 1999. Extended Core Model for Representation of the Common Reference Model for Procedures. GALEN-IN-USE Project Deliverable. Revised for OpenGALEN October 1999. Nijmegen: OpenGALEN.

Ruan, W., Bürkle, T., Dudeck, J. 2000. A Dictionary Server for Supplying Context Sensitive Medical Knowledge. In: Overhage, M. J. (ed.) *Converging Information, Technology, and Health Care: Proceedings of the 2000 AMIA Annual Symposium*. Nov 4-8, Los Angeles, California. p. 719-723. Bethesda (Maryland): American Medical Informatics Association.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. 1993. Objektorientiertes Modellieren und Entwerfen. Munich: Carl Hanser.

Rumbaugh, J., Jacobson, I., Booch, G. 1999. *The Unified Modeling Language Reference Manual*. The Addison-Wesley Object Technology Series. Reading (Massachusetts): Addison-Wesley.

SNOMED International 2004. SNOMED International, a division of the College of American Pathologists (CAP) [homepage]. Cited 13.04.2004. Available from: http://www.snomed.org

Spackman, K. A. 1999. Terminology Convergence: SNOMED Gets a Boost. *MD Computing* 16(5)**:**23-25.

Spackman, K. A. 2001. Normal Forms for Description Logic Expressions of Clinical Concepts in SNOMED RT. In: Bakken, S. (ed.) *Visions of the Future and Lessons from the Past: Proceedings of the 2001 AMIA Annual Symposium*. Nov 3-7, Washington DC, USA. p. 627-631. Bethesda (Maryland): American Medical Informatics Association.

Spackman, K. A., Campbell, K. E. 1998. Compositional Concept Representation Using SNOMED: Towards Further Convergence of Clinical Terminologies. In: Chute, C. G. (ed.) *A Paradigm Shift in Health Care Information Systems: Clinical Infrastuctures for the 21st Century: Proceedings of the 1998 AMIA Annual Symposium*. Nov 7-11, Orlando, Florida, USA. p. 740-744. Philadelphia: Hanley and Belfus.

Spackman, K. A., Campbell, K. E., Coté, R. A. 1997. SNOMED RT: A Reference Terminology for Health Care. In: Masys, D. R. (ed.) *The Emergence of Internetable Health Care. Systems that Really Work: Proceedings of the 1997 AMIA Annual Symposium*. Oct 25-29, Nashville, USA. p. 640-644. Philadelphia: Hanley and Belfus.

Stearns, M. Q., Price, C., Spackman, K. A., Wang, A. Y. 2001. SNOMED Clinical Terms: Overview of the Development Process and Project Status. In: Bakken, S. (ed.) *Visions of the Future and Lessons from the Past: Proceedings of the 2001 AMIA Annual Symposium*. Nov 3-7, Washington DC, USA. p. 662-666. American Medical Informatics Association.

van Bemmel, J. H., Musen, M. A. (eds.) 1997. *Handbook of Medical Informatics*. Heidelberg: Springer.

Welty, C. A., Ferrucci, D. A. 1994. What's in an Instance? Technical Report No. 94-18. Troy (New York): Computer Science Dept., Rensselaer Polytechnic Institute.

WHO (ed.) 1992. *International Statistical Classification of Diseases and Related Health Problems, Tenth Revision*. Vol. 1. Geneva: World Health Organization.

Index of Figures

Figure 1.1 Overview of the model of the Data Dictionary Version 1. 5
Figure 2.1 Monohierarchy vs. polyhierarchy...11
Figure 4.1 Examples of relations in the UMLS. 40
Figure 5.1 Shades of color used in the UML diagrams. 50
Figure 5.2 Data types defined for the DD model. 51
Figure 5.3 The central classes of the DD model. 52
Figure 5.4 Comparison of the representation of terms in the naive approach and in the DD model. 54
Figure 5.5 Validity of definitions. 57
Figure 5.6 Structure of definitions: informal and semiformal definitions. 60
Figure 5.7 Illustration of the relation between a category and its template consisting of template components. 61
Figure 5.8 Activity diagram for creating a new instance of `DefinitionVersion`. 63
Figure 5.9 Illustrating the connection between templates and filled templates. 64
Figure 5.10 Bibliographical annotations of definitions. 65
Figure 5.11 Representation of semantic relations. 66
Figure 5.12 Examples of instances of `Relationship` and `RelationalRole`....................67
Figure 5.13 Example of the definition of multiplicities. 68
Figure 5.14 Types of relational links. 70
Figure 5.15 Examples of strings, terms, and concepts illustrating different notions of ambiguity. 73
Figure 5.16 Representation of communities and content areas. 74
Figure 5.17 Illustration of a community hierarchy. 75
Figure 5.18 Model classes and associations for representing contexts. 82
Figure 5.19 Excerpt from the DD model providing an overview of the classes and associations relevant to the resolution of strings to concepts. 87
Figure 7.1 Architectural design illustrating the distinction between the DD-Server and the Onto-Builder. 97
Figure 7.2 Screenshot of the Protégé application with the newspaper example from the Protégé tutorial opened. 100
Figure 7.3 The “Forms Tab”. 101
Figure 7.4 Screenshot of the Onto-Builder prototype implemented in Protégé..................104

Index of Tables

Table 4.1 Excerpt from ICD-10. 31

Table 4.2 Examples of LOINC codes...36

Table 4.3 Excerpt from the Semantic Network hierarchy. 39

Table 5.1 The reasons for the occurrence of differences...49

Table 6.1 Overview of the requirements and their fulfillment within this thesis. 90

Table 6.2 Overview of DD model elements for the representation of differences in definitions...92

Table 7.1 Summary of the mapping of those classes relevant to implementing the DD template mechanism on the basis of Protégé’s form mechanism. 106

Table 7.2 Summary of the mapping of the associations relevant to implementing the DD template mechanism on the basis of Protégé’s form mechanism. 107

Table 7.3 Summary of the mapping of the class `Relationship` and related classes, which are also implemented on the basis of Protégé’s form mechanism. 107

Appendix A – Specification of View Computation

The information given in this Appendix describes the details of the view computation explained in section 5.7. For comprehensibility, the computation of the view axes is described in OCL-like pseudocode, since the complete OCL formulation is more textual at some points and is thus less readable[71].

The dot-notation of OCL for navigating between classes along associations between them is introduced in section 5.2. Variables are specified according to OCL, e.g. `U_Communities[1..*]: Set(Community)` defines the variable `U_Communities` which is of type set containing as elements at least one instance of the class `Community`. Further, although in OCL the same symbol (=) is used for equivalence as well as for value assignments, we use "`:=`" for the latter and "=" only for equivalence in order to avoid confusion.

User Properties

User properties are used as default values of certain context properties (as specified in the respective table on the view axes).

user property	description	definition of the user property (specifies how to query the DD model for the user property)

U_Communities	user is a member of communities	**context** User
		def: U_Communities[1..*]: Set(Community) :=
		  self.allCommunities

U_Areas	user has access to content areas	**context** User
		def: U_Areas[1..*]: Set(ContentArea) :=
		  self.hasAccessTo.contentArea

U_AreaRoles	in each content area, the user fills certain roles	**context** User
		-- a set of tuples (community, user roles)
		def: U_AreaRoles[1..*]: Set(TupleType(
		   area[1]: ContentArea,
		   roles[0..*]: Set(Role))) :=
		     self.hasAccessTo.contentArea->collect(a |
		     Tuple {area: ContentArea := a,
		           roles: Role := a.hasAccessTo.role})

U_Competences	the user is expert in a set of domains	**context** User
		-- trace the association hasCompetences towards Domain
		def: U_Competences[0..*]: Set(Domain) :=
		  self.domain

[71] e.g. an intersection of two collections is written in OCL as: `collection1->intersection(collection2)`, while it is abbreviated in the view specification as: `collection1 ∩ collection2`

Overview of View Axes and their Dependence on Context Properties

view axis	description	depends on **context property** (=parameter of view axis)	default value of context property	constraint on context property
`V_ContentArea`	restrict view to certain content areas	`C_Areas[1..*]: Set(ContentArea)`	– Although it may seem possible to use `U_Area` as default value, this is not sensible because `U_Area` contains exactly those content areas that the user has access to. Therefore, using this as value for the view axis would not have any effect on the content which is visible.	`C_Areas ⊂ U_Areas`

`V_organizational-Validity`	restrict view to content which is valid in certain community/-ies (organizational context)	`C_Communities[1..*]: Set(Community)`
		`C_OrgValString[0..1]: String`
			`U_Communities`
			–
				–
				–

`V_geographical-Validity`	restrict view to content which is valid in certain geographical areas (geographical context)	`C_GeogrAreas[1..*]: Set(GeographicalArea)`	–	–
`V_RoleRelevance`	restrict view to content which is marked as being relevant to certain role(s) (in each content area `area`, the user fills certain `roles`)	`C_AreaRoles[1..*]: Set(TupleType( area[1]:ContentArea, roles[1..*]:Set(Role) ))` Note that the multiplicity of roles must be at least one (i.e. `[1..*]`, instead of `[0..*]` as in `U_AreaRoles`) because the view selects content which is relevant to the specified roles only. Thus, if no roles are selected, the view will be empty.	`U_AreaRoles` (only possible if its multiplicity conforms to that of `C_AreaRoles` as explained on the left)	**if** `C_Areas` is set **then** `C_AreaRoles` must specify roles for each content area in `C_Areas` **else** `C_AreaRoles` must specify roles for each content area in `U_Areas`

view axis	description	depends on **context property** (=parameter of view axis)	default value of context property	constraint on context property
`V_degreeOf-Difficulty`	restrict view to content at an appropriate degree of difficulty (the user is competent in the specified set of domains (“expert”) and is layman in all other domains → expert or layman definitions and terms are displayed according to the user’s competence in the concept’s domain)	`C_DomainCompetences [0..*]: Set(Domain)` The empty set is allowed as a context parameter value since this means that the user is layman in all domains.	`U_Competences`	–

`V_Category`	restrict view to concepts of certain categories and – if `showFullDef=false` – only show category part of the semiformal definitions (informal definitions are always completely included if the concept is included)	`C_Categories[1..*]: Set(Category)`
		`showFullDef: Boolean`
			–
			`true`
				–
				–

`V_Domain`	restrict view to content of certain domains	`C_Domains[1..*]: Set(Domain)`	–	–
`V_Language`	restrict view to content of a specific language	`C_Language[1]: ContentLanguageT`	–	–

In section 5.7.1, it is outlined how to create coherent subsets of the DD contents. This comprises two steps: firstly, an initial selection of instances according to a view axis, and secondly, the inclusion of further instances reachable from those selected such that there exist no isolated instances in a view.

Below, the first of these steps is specified for each view axis in OCL-like pseudocode. The statement `include` is used to express that some instance is included in a view, while `exclude` means that some element which was included in a previous step of the computation is excluded again.

Beforehand, though, with regard to the second step, the classes which are affected by the view computation (i.e. the view mechanism is only concerned with selecting subsets of instances of the listed classes, but for example not with selecting categories, domains, etc.) as well as the associations between these classes which are to be traced in the process of determining "reachable" instances are listed. A more formal specification of the second step is not given since it is difficult to elaborate this without an implementation, and would further be very space-intensive.

Classes: `String`, `Term`, `Concept`, `Definition`, `SemiformalDefVersion` `InformalDefVersion`, `Translation`, `FilledTemplate`, `RelationalFilledComp`, `AttributiveFilledComp`, `TextualFilledComp`

Associations: `designates`, `abbreviations`, `lexicalVariants`, `preferredString`, `preferredTerm`, `TermValidity`, `hasVersions`, `alternativeDefinitions`

In many view axes, we need to compute the transitive closure of a set of instances of a class with regard to an association which defines an acyclic directed graph. For example, the transitive closure for class `Community` (cf. the explanations about the community hierarchy in section 5.6) is computed as follows.

```
transitiveClosureComm(s:Set(Community)):Set(Community) {
  s_old:Set(Community) := ∅
  s_new:Set(Community) := s
  -- find all including communities
  while s_new ≠ s_old do {
    s_old := s_new
    for each x:Community ∈ s_old do {
      y:Set(Community) := x.includingComm
      s_new := s_new ∪ y
    }
  }
  -- eliminate those communities not in s with
  -- transitiveValidity=false
  s_new = ∅
  for each x:Community ∈ s_old do {
    if (x ∈ s) ∨ (x.transitiveValidity = true)
    then s_new := s_new ∪ {x}
  }
  return (s_new)
}
```

A similar, but simpler computation (without the additional attribute `transitiveValidity`) is also necessary for the classes `GeographicalArea` and `Domain`, as specified below.

```
transitiveClosureGeoArea(s:Set(GeographicalArea)):Set(GeographicalArea)
{
  s_old:Set(GeographicalArea) := ∅
  s_new:Set(GeographicalArea) := s
  -- find all including areas
  while s_new ≠ s_old do {
    s_old := s_new
    for each x:GeographicalArea ∈ s_old do {
      y:Set(GeographicalArea) := x.includingGeoArea
      s_new := s_new ∪ y
    }
  }
}
```

```
transitiveClosureDomain(s:Set(Domain)):Set(Domain) {
  s_old:Set(Domain) := ∅
  s_new:Set(Domain) := s
  -- find all superdomains
  while s_new ≠ s_old do {
    s_old := s_new
    for each x:Domain ∈ s_old do {
      y:Set(Domain) := x.superdomain
      s_new := s_new ∪ y
    }
  }
}
```

These three auxiliary methods are employed in the specifications of the view axes given below.

V_ContentArea(C_Areas)

```
for each Concept c do {
  if c.contentArea ∈ C_Areas
  then include c
}
```

V_geographicalValidity(C_GeogrAreas)

```
for each DefinitionVersion d do {
  if d.geographicalArea[0..*] ∩
     transitiveClosureGeoArea(C_GeogrAreas) ≠ ∅
  then include d
}
```

V_organizationalValidity(C_Communities, C_OrgValString)

```
for each DefinitionVersion d do {
  if OrgValString = ""
  then {
    if d.community[0..*] ∩ transitiveClosureComm(C_Communities) ≠ ∅
    then include d
  } else {
    if (d.community[0..*] ∩ transitiveClosureComm(C_Communities ≠ ∅))
        ∧ (d.orgValidityComment = C_OrgValString)
    then include d
  }
}
```

V_Category(C_Categories, showFullDef)

```
for each Concept c do {
  if c.category[0..*] ∩ C_Categories ≠ ∅
  then  {
    include c

    if showFullDef = true
    then {
      -- all definitions are included completely
      include c.definition.definitionVersion
    } else {
      -- only include the category's template part of
      -- the semiformal definitions
      flag = false
      for each c.definition.SemiformalDefVersion dv do {
        for each dv.filledTemplate fTemp do {
          if fTemp.template.category ∩ C_Categories ≠ ∅
          include fTemp ∧ (flag := true)
        }
      }
      -- Exclude c if none of its filledTemplates was included
      -- in the previous step. This can happen if the desired
      -- category template is optional and is not filled by
      -- any definition of the concept
      if flag = false exclude c
    }
  }
}
```

```
V_RoleRelevance(C_AreaRoles)

for each Concept c do {
  -- define helper variable which contains the roles
  -- for which the concept is relevant
  def conceptRelevance: Set(Role) := c.role[0..*] ∪
        c.category.role[0..*] ∪ c.domain.role[0..*]
  -- define helper variable which selects from C_AreaRoles the
  -- user's roles in the content area to which the current
  -- concept c belongs
  def currAreaRoles: Set(Role) :=
        (C_AreaRoles->select(aR | aR.area := c.contentArea).roles)

  -- compute view
  for each c.definition.definitionVersion dv do {
    if dv.oclIsTypeOf(InformalDefVersion)
    then {
      -- include the whole definition if the concept is relevant
      if (conceptRelevance ≠ ∅) ∧ (conceptRelevance ∩
          currAreaRoles ≠ ∅)
      then include dv
    }
    else {
      -- i.e. it is a SemiformalDefVersion -> test each
      -- filled template component separately for its relevance
      d:=dv.oclAsType(SemiformalDefVersion)
      for each d.filledTemplate.filledComponent fComp do {
        -- define helper variable which contains the roles for
        -- which the component is explicitly relevant
        def componentRelevance: Set(Role) :=
               (fComp.templateComponent.role[0..*] ∪
                fComp.role[0..*])
        if componentRelevance ≠ ∅
        then {
          if componentRelevance ∩ currAreaRoles ≠ ∅
          then include fComp
        } else {
          -- i.e. no role is specified for the
          -- component -> test for the concept's relevance
          if (conceptRelevance ≠ ∅) ∧ (conceptRelevance ∩
              currAreaRoles ≠ ∅)
          then include fComp
        }
      }
    }
  }
}
```

V_Domain(C_Domains)

```
for each Concept c do {
  if ( c.category.domain[0..*] ∪ c.domain[0..*] ) ∩
       transitiveClosureDomain(C_Domains) ≠ ∅
  then include c
}
```

V_degreeOfDifficulty(C_DomainCompetences)

```
for each Concept c do {
  for each c.term t do {
    if c.domain[0..*] ∩
       transitiveClosureDomain(C_DomainCompetences) ≠ ∅
    then {
      if t.degreeOfDifficulty ∈ {'expert', null}
      then include t
    } else {
      if t.degreeOfDifficulty ∈ {'layman', null}
      then include t
    }
  }
  for each c.definition.definitionVersion d do {
    if c.domain[0..*] ∩
       transitiveClosureDomain(C_DomainCompetences) ≠ ∅
    then {
      if d.degreeOfDifficulty ∈ {'expert', null}
      then include d
    } else {
      if d.degreeOfDifficulty ∈ {'layman', null}
      then include d
    }
  }
}
```

V_Language(C_Language)

```
for each Concept c do {
  for each c.term t do {
    if t.language=C_Language
    then include c.term
  }
  for each c.definition.definitionVersion d do {
    -- retrieve the DefinitionVersion in the required language
    -- (result may be empty if d does not exist in that language)
    include d(language)
  }
}
```

As a last note, it should be mentioned that most of the context properties contain sets of instances, e.g. sets of domains that are to be included in the view. All the view axes' specifications given above treat the values as being connected by "OR", e.g. a concept is included if its set of domains and that of `C_Domains` is overlapping. It is, of course, also conceivable that a user wants to include only those concepts which match the specified set of domains directly (i.e. being connected by "AND"). Moreover, there is the issue of null values, since it is for example optional to specify values for a concept's domain. The implementation of the view mechanism could thus provide additional parameters which allows a user to choose between the AND and OR variant as well as between including or excluding null values. We have refrained from including these parameters in the specification of the view mechanism above in order not to complicate it further.

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ort, Datum   Unterschrift