**Parthenos Project -**
ICS-FORTH (Foundation for Research and Technology – Hellas)
Institute of Computer Science
N. Plastira 100
Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

**Summer**

17

08

**Fall**

Principles for Modelling Ontologies: A Short Reference Guide

Parthenos / ICS-FORTH

This document is a first draft of an ontological modelling reference booklet desgined to aid modellers developing or extending ontologies intended to support data integration for an empirical research domain

Editor: George Bruseker

Contributors: Francesco Beretta, Maria Daskalaki, Emiliano Degl’Innocenti, Martin Doerr,

Øvyind Eide, Achille Felicetti, Anaïs Guillem, Dimitris Kafetzopoulos, Fahad Khan, Carlo Meghini, Nicola Spyratos, Maria Theodoridou, Athanasios Velios

Date of Creation: July, 2017

Last Update: September, 2017

Version 0.2

Acknowledgements: Parthenos, FORTH + more

**Table of Contents**

[**General Introduction**](#_nk4zxwbnppug) **4**

[Theoretical Background](#_27oh55vf3a33) 5

[**Process Model**](#_ne4q5se3pgea) **10**

[Phase A: Purpose Definition](#_lchdal8qwj8y) 11

[Phase B: Ontology Constructs Definition](#_xiyfmxpz7iel) 12

[Phase C: Implementation and publishing](#_kbtn6q57cnyj) 14

[Mapping](#_fmtqya6nyde) 14

[**Principles Introduction**](#_30j0zll) **18**

[**Glossary**](#_3fv7ayo91pjc) **20**

[**Engineering from an Empirical Base**](#_1fob9te) **25**

[1.1 Model from existing / actually used structured information sources (whenever available)](#_2et92p0) 26

[1.2 Model according to the research questions justifying the structured information](#_tyjcwt) 27

[1.3 Model from actual information values](#_3dy6vkm) 28

[**Knowledge Structure Basics**](#_1t3h5sf) **29**

[2.1 Detect hidden relations in terms](#_2s8eyo1) 30

[2.2 Distinguish particulars from universals in the target domain](#_3rdcrjn) 32

[2.3 Do not define the same property twice for different classes. Find the superclass for it](#_26in1rg) 33

[2.4 IsA is an increase of instances and a decrease of properties](#_lnxbz9) 34

[**Concept Relevance**](#_r6wxve9igk3t) **36**

[3.1 Model primitive concepts first](#_1ksv4uv) 37

[3.2 A class should allow the formulation of a query that answers a relevant question](#_44sinio) 38

[3.3 Model manageable units](#_z337ya) 39

[3.4 Model concepts that express the least interpretational position in order to make the model robust against revision](#_1y810tw) 41

[**Open World**](#_4i7ojhp) **42**

[4.1 Never define a class as complement](#_2xcytpi) 43

[4.2 Cover incomplete details of knowledge by what you do know](#_1ci93xb) 44

[4.3 Do not create closed worlds of properties](#_3whwml4) 45

[**Open World and Knowledge Progress**](#_2bn6wsx) **47**

[5.1 Support progressive improvement of classification knowledge by IsA hierarchy](#_qsh70q) 48

[5.2 Do not model conclusions before and without their reasons](#_3as4poj) 49

[5.3 Describe the intension of and declare classes that model the parts of the domain you understand](#_1pxezwc) 50

[5.4 Model domains and range or properties consistent with your level of knowledge of the domain of discourse](#_49x2ik5) 51

[**Open World and Knowledge Base**](#_2p2csry) **52**

[6.1 The absence of a property in the knowledge base is not its negation in reality](#_147n2zr) 53

[6.2 Allow alternatives or contradictions in the data](#_3o7alnk) 54

[6.3 Make sure alternative assertions can be unambiguously related to a single entity](#_23ckvvd) 55

[6.4 Explain Data Structures](#_ihv636) 56

[**Objectivity**](#_32hioqz) **57**

[7.1 Be view neutral](#_41mghml) 58

[7.2 Avoid concepts depending on a personal/ spectator perspective](#_2grqrue) 59

[7.3 Avoid concepts depending on accidental and uncontextual properties](#_vx1227) 60

[7.4 Maintain independence from scale](#_3fwokq0) 61

[**Language and Concepts**](#_1v1yuxt) **62**

[8.1 Don’t confuse polysemy with multiple abstractions](#_4f1mdlm) 63

[8.2 Most binary relationships in intuitive conceptualizations conceal temporal entities](#_2u6wntf) 64

[**Conceptual Modelling Checklist**](#_19c6y18) **65**

#

# General Introduction

This document is a reference guide to the ontology engineering method underlying the construction of the CIDOC CRM (ISO21127:2014) and its extensions. It is not intended as self-teaching material or as an introduction to newcomers in the field. It is intended as an accompanying handbook for users that have received respective training by courses or other didactic material and are in a process to use the CIDOC CRM or its extensions, to map legacy data structures to it, or to develop further extensions. It aims at being as compact and comprehensive as possible.

Formal ontologies are a means to describe information structures, such as database schemata, XML Data Type Definitions, etc., also called “data models”. They were developed by computer scientists in collaboration with philosophers from the mid 1990s on, in order to overcome the idiosyncratic and incompatible data models developed by independent teams for the same domain. In order to overcome this problem, the idea was and is to specify the relevant kinds of things (objects, people, places, events, time, concrete ideas etc.) and their possible relationships in the part of the universe covered by one or more information systems with the use of logical expressions.

This specification aims at representing exclusively the relevant possible states of affairs as currently commonly understood by domain experts, creators, and users of information systems. It must not compromise the correctness of description of the universe of discourse (“ontological commitment”) to:

(1) the efficiency of data storage and processing of particular implementation platforms;

(2) particular language and terms of subdomains;

(3) simplifications of data entry by omitting case-specific background constants and default deductions; and

(4) the simplicity of querying data for coarse and indirect relationships (such as “provenance of an object”).

Only under the above four conditions the formal ontology can commonly be accepted and will be general enough to overcome idiosyncratic constructs. The consequence of this necessary restriction is that data entry forms and their labels, query formulation tools and their terms, and system internal methods of storing and implementation will and must in general deviate from the formal ontology. So long as these “deviations” can be described in terms of logical relations of the common formal ontology to the constructs in the implemented components, the formal ontology is operational and effective. Its use will enforce a good practice of information modelling and will support the integration of different systems of overlapping domains and the migration of data to new and more powerful platforms.

The confusion of the above four technical requirements with the effective principles of ontology engineering as described in this guide is a major error of so-called “practitioners” and practical obstacle to find the effective constructs. Unfortunately, in order to keep this guide compact enough, we could not include a discussion of the current methods to make data entry and querying simple and implementations fast and efficient on top of a complex ontology, in order to make the reader of this guide confident that these issues can be technically solved. We can only point to respective literature. Lack of understanding of this issue has fostered the idea that virtually any data structure is an ontology, in particular a schema in the form of known “knowledge representation languages”, which throws us back in the 2010s to the idiosyncrasy of local systems this method aimed at overcoming in the 1990s. We do not support this opinion out of ample practical experience, and we do not regard “vocabularies” and terminological systems “ontologies”.

This guide describes a methodology for formal ontology development presented in the form of a simple process model of recommended steps and a series of distinct principles for consultation during conceptual modelling activities. These principles are deemed pertinent to the creation of information systems about scientific and scholarly knowledge using formal ontology, although in practice the scope of their application may be broader. The principles are arranged in thematic groups representing clusters of relevant, practical modelling issues and responses. They constitute a list of functional conclusions regarding key questions/problems recurrently encountered in conceptual modelling activities that have been arrived at through over twenty years of hands on, empirical engagement in ontology construction.

## Theoretical Background

The principles presented here are a consolidation of a consciously different perspective on formal ontology development to other major methodological approaches that have been outlined in the literature. This differentiation arises out of the practical effort to apply existing methodologies and finding the need to take new ground positions with regards to the task of ontology development itself and the limits that it faces in order to meet the end goal of providing functional models for the fields under investigation. It is necessary, then, in preface to an explication of the proposed process model and principles, to elaborate on the background assumptions and context from which they have been developed.

This introduction briefly outlines how the ground positions on formal ontology that we take here differ with regards to some other major schools of formal ontology, leaving detailed argumentation defending specific points to be fully addressed in other fora.

The basic issue to address is what the product of a modelling effort is: what is a formal ontology? There is general agreement between different schools of modelling at least that a formal ontology is a tool for communication in distributed information environments. We, further, take the position that with regards to scientific and scholarly communication, the model is also necessarily a means of representation of some aspects of reality.

Where there is room for disagreement and position-taking after this relates to the issues of:

1. what kind of representational tool a formal ontology is;
2. the precision of the relation between a formal ontology and the reality it attempts to represent; and
3. what aspect of reality a formal ontology does represent and how this affects its verifiability.

We argue that formal ontologies with regards to their form belong to the class of logical theories and, as such, belong to the general class of discrete mathematical theories. They are further distinguished by explicitly aiming to approximate reality and, therefore, are conditioned by this representative functionality – i.e. their connection to empirical reality - with regards to their truth functionality The kinds of mathematical representation of reality include but are not limited to such different forms as continuous functions, neural networks, statistical models, and logical theories.

Scientists and many scholars make and use mathematical models of reality in order to communicate about it or to predict its behavior. The suitability and application of these models varies substantially depending on: i) the nature of the aspect of reality to be modelled, ii) the cognitive capacity of some group to perceive and represent the reality intended by the model, and iii) the features of the aspect of reality modelled which are of interest for scientific prediction/communication.

Thus far, the position presented in relation to (a) should be largely in accord with the approaches of other schools of formal ontology. Where our position begins to differ lies in the conclusion we draw from this categorization based on our understanding of the basic limits of mathematical models as representational tools.

The representational success of any mathematical model referring to reality presupposes that some human beings (scientist, scholars and other users) are able to understand and then instantiate said model with relevant parameters taken from reality and, thereafter, can compare results of data represented in the model with actual situations in reality (this observational checking can as much be carried out through the natural senses as aided by artificial sensors) - and/or amongst themselves. Crucially, this entails that the precision of applying mathematical models to reality is ultimately externally limited by the precision of the requisite parameter provision (data collection) and result comparison by knowing agents. This accuracy is necessarily empirically affected by the factors of: x) determinacy of the defined parameters in the reality itself[[1]](#footnote-1) , y) precision of the tool used to observe it, z) the precision of the interpretive frame operative to give sense to the results/data.

Given the above, we take a different position on issue (b) than schools that would argue that a well-built ontology should ideally hold a one-to-one relation between its concepts hand the world it aims to represent. We argue, instead, that if we accept the representational mode of the ontology as a logical theory, this introduces necessarily a limit to the things it can represent at all as well as the potential precision of its representation.

First of all, a formal ontology is not suitable for representing all aspects of reality that can be named. Because of its representational form, it is capable of being representational only with regards to clearly identifiable individual items, out there in reality, with relations that can – in principle - be verified or falsified as holding or not by independent persons. Examples of these would be cars, hammers, a shoe maker, or a village, but would not include other aspects of reality such as “a” cloud, wave, or wind, all of which are better represented by other models suitable to representation of continuous phenomena.

Second, the representational precision of an ontology is limited by its form itself. Classes and relations specified at the level of the universal are always approximations for a sphere of reality. They cannot be expected to form an exact duplicate of the reality modelled. The inherent inability of an ontology to provide a one-to-one ideal surrogate for real world objects in no way, however, makes it useless. Rather, we maintain that the ontology development process must start with identifying, if the degree of deviation of reality from each element of the model is tolerable for the intended communication purpose of the target community both in terms of precision and in terms of statistical occurrences of exceptions. If this is established, then logical rules for reasoning can be formulated and used as an additional engineering construct.

That being said, the lack of a one-to-one relation between a model and reality does not entail a subjectivist position on (c). Such a position is staked out by some schools of conceptual modelling[[2]](#footnote-2) that limit the goal of modelling of a formal ontology to replacing intuitive conceptualizations held by groups of actors by logical terms, arguing that how such representations then relate to the reality as such constitutes a black box outside the scope of modelling activity. That is to say that for this school of thought, delving into the questions of cognitive science and how correspondence is established between the knower and some known thing is deliberately left outside of the questions posed by the conceptual modeller. This position is taken out of a principle of caution and as a practical measure to limit the scope of formal ontology to a reasonable scale, unburdening it from a direct scientific activity it is not responsible for.

To our experience, however, this self-restriction is both unnecessary and, more importantly, unhelpful to good modelling. Reference to reality is both a necessary and grounding factor for good modelling and leaving it out of scope leaves models without an objective referent. Such a limitation is also not justified from a pragmatic, epistemic perspective. There exist many things in our reality and everyday life that people can reliably identify, classify, relate, and communicate in a form compatible with formal ontologies without using logical definitions. This is evidenced, within pragmatic epistemic conditions, by the fact that we survive in complex environments and that this is the legal basis for the normal liability of people for their actions in a complex social environment (despite exceptions, border cases, and limited precision of expression). Therefore, reference to reality is functional in an engineering sense.

Such reference to reality is of course a function of highly complex factors, studied by cognitive science, of how communicative agents adjust to contexts of reference and subconsciously modify concepts in a dynamic and adaptive way and yet are still able to communicate. This flexibility and sophistication is something that formal ontologies and information technology do not have. The context of reference consists of a combination of the phenomena under consideration (for instance being in a clinic or on the highway) and the questions a communication tries to answer. We maintain that it is the richness of knowledge behind “surface concepts” and the flexibility to adapt concepts to a changing reality, and not the deficiencies of knowing agents to formulate logically cogent conceptualizations, that hinders the immediate transfer of human concepts into an information environment.

For instance, most people are well aware of the details of how a human being comes into existence. However, depending on the question, people will choose to regard it as having begun with conception or at birth. They may regard cut hair as a body part or as waste. This is not due to subjectivity or different perceptions of reality. They represent, rather, a function of selecting a suitable simplification of reality to communicate answers to queries such as “when can a human being become heir of someone” or “when should we protect human life”. Even within the same “domain” such multiple definitions may occur.

Consequently our ontology engineering methodology consists of:

1. Fixing a context of discourse with a limited subject matter and limited generic questions in order to reduce the interpretative complexity of concepts, a sort of “requirements specification”.
2. Exploring which concepts in this context can be formulated as a logical theory that can sufficiently represent the reality under consideration and support the intended generic questions.

This is done by selecting or inventing concepts from the range of experience of experts, from documentation, and by actively learning about phenomena in the aspect of reality under investigation. Since human experience is not ready-to-hand, but partially subconscious, and individual and collective knowledge is limited, this process of “knowledge engineering” consists of making participating experts consciously aware of the relevant reality, and facilitates them to learn from each other, from documentation and even active observation about exceptions, border cases etc.

Since this is a process of systematically increasing collective knowledge, it implies that the basis for a formal ontology (that has the required quality of representing reality) is a limited set of knowledge, which is expected to grow, but never be exhaustive. Therefore, the methodology contains a set of innovative advice on how to foresee the effect of new facts and to evolve the ontology from “safe grounds of knowledge”. The so-called “Open World Assumption” is such a principle known from computer science, which we generalize here for that purpose.

For the above reasons, the following guide proposes a process we found apt to help elicit systematically the relevant knowledge and to generalize it and widen its scope. It also proposes a set of principles to be applied alongside this process that aim to facilitate the work of a team of conceptual modellers whose goal is to build a functional formal ontology useful for some empirical domain and group of users.

# Process Model

In the following we give a compact description of the CRM ontology engineering process, as it has emerged from over twenty years of practice. This process is similar but substantially different from processes that have been described going back to Booch and Coud-Jordan for conceptual modelling with object-oriented programming languages, and which often have uncritically been reapplied to ontology engineering. The major differences are (a) that we find classes for the relevant properties instead of properties for relevant classes, and (b) that the process is completely iterative. Another substantial difference is that we deal only with ontologies made to formulate propositions in information systems, as originally described by Thomas Gruber [XXX]. We do not apply this methodology to create terminological systems (vocabularies, thesauri, classification systems, typologies, etc.) to be used as data in propositions in the target systems, such as the AAT or LCSH.

In the process of ontology engineering under the above restriction we can distinguish two different starting situations, referred to throughout the rest of this guide as Case A and Case B:

1. building an ontology “from scratch” for a new domain, or
2. building an ontology from a functional set of information structures in use for a domain.

In both cases, we of course assume (i) that the engineer is aware of the CIDOC CRM as a set of concepts for reuse in the process we describe below, and (ii) that the engineer will recognize if a concept emerging in the process has already been described more or less in the CRM or an extension of it, or in another ontology. However, this must not create a bias to “press” things under CRM concepts, but rather be each time a test of the adequacy of the CRM. Some concepts in the CRM will most likely be generic enough to be adequate for any ontology dealing with the past. For some other concepts a slight expansion of its definition can be functional, as long as that does not create other inconsistency in the CRM. Otherwise, new concepts should be created without any bias with regards to existing ones. Therefore, we describe the process initially in a way as if no prior ontology exists, followed by describing the process of actually “mapping” to the CRM, i.e., fitting concepts under the CRM, as a special aspect of the general engineering process.

**The Process**

This process is iterative. Any step may be occasion to reconsider previous ones. Each iteration improves the overall understanding. Care must be taken to recognize and break circular arguments.

The process consists of 3 phases:

1. Definition of purpose
2. Ontology construction
3. Implementation and publishing

## Phase A: Purpose Definition

The steps are:

* 1. **Definition of a theoretical scope**, e.g., : “political history”, “recording art conservation”, “history of music and performances” etc. This restricts the kind of phenomena under consideration to things studied or to be studied by one or more explicit disciplines.
	2. **Definition of the overall question** to an information system, for instance “impact of technology on material conditions of living”, or “evidence for archaeological opinions documented in the excavation process”. It is good practice to explicitly exclude topics in terms of overly details or rareness. Each relation to be defined in the next phase can be regarded as element in a formalized question, a so-called query, which should contribute to the overall question according to the experts’ opinion.
	3. **Definition of empirical source material** (“practical scope”) to be used to elicit the relevant concepts. These can include texts, images, videos, datasets, and information structures. Information structures must be accompanied with representative datasets using them.
		+ Case (1): the scope of the target ontology is **limited** to what is relevant in and meant by these sources. E.g., FRBRoo is limited to what is directly or indirectly meant by FRBR documents – texts defining data structures for future use.
		+ Case (2): The sources are a **representative** selection with sufficient coverage and may later be extended. E.g., CRMbasic was initially engineered from museum collection management systems.
		+ In Case A, building an ontology from scratch, it is advisable that a team of experts extracts intuitive entities and relations from a reasonable set of sources, typically texts, and uses them as surrogate for existing data structures that relate to the overall questions.
		+ In Case B, building an ontology from a particular set of used information structures, their successful use is taken as evidence for the relevance of the underlying concepts. For example, CRMbasic is limited to senses directly or indirectly meant by existing and used information structures (we will call the elements of preexisting information systems also “intuitive terms” without implying a prejudice).

## Phase B: Ontology Constructs Definition

The steps are:

1. **Take a list of intuitive, specific terms** from the empirical source materials. These can be relations, entities from information structures, or categorical terms in the narrower sense. Even though looking for properties and relations is reserved for the next step, it is difficult to imagine relations without conceptualizing the entities they relate. However, it is advisable to avoid making abstract generalizations in this step because the properties are best understood in very specific cases. In information structures, strings and Boolean values may hide concepts!
2. **Create a list of properties for the intuitive entities.**  Find their relevant properties (behavior) for the discourse about the intended overall questions. Discuss all known reality within the scope. Each property must answer a specific question that helps understanding the overall questions. Detect and resolve polysemy by splitting terms into multiple concepts if necessary (“Where was the university when it decided to take more students?”). Change mentally the context of use of a term and observe its properties (e.g., is “pencil” a name of an object in a museum and in a shop?).

1. **Detect entities hidden in intuitive relations**. These are most frequently events, activities that are either shortcut by a (binary) relation or that initiate or terminate a relation. This may reveal that some properties from step 2 are not relevant, in which case, revise step 2. There may be source material supporting the hidden entities, in which case, revise step 1 and possibly Phase A. Truly n-ary relations are relatively rare (e.g., relative positions), but if encountered they must also be modelled as entities in the current knowledge representation languages.
2. **Detect classes from properties.** In this step, use only properties that are not logical deductions from other properties (e.g. « creator of » is a deduction from « created by – activity – carried out by »). Find the general classes for which each property is characteristic. In other terms, find the one most specific class that generalizes over all classes for which you are sure that the property applies as domain or range.
3. **Provide identity conditions to the classes.** Answer the questions:
	* By what something be determined as instance of this class?
	* Is there something that is and that definitely is not an instance of this class?
	* What makes an instance distinct from another and be the same after some time?
	* What belongs to it as extent or part?
	* How do instances come into being/ end being?

Discuss all known reality within the scope. Seek expert knowledge about exceptions. Find out if this concept has been thought of or been defined somewhere. Try to learn or adopt better definitions.

1. **Create the class hierarchy.** Determine which of the new classes are superclasses of others, i.e., if a domain (range) of a property A as defined above generalizes (possibly besides others) over a domain (range) of a property B. In other words, wherever B applies, A applies, but not vice-versa. This process may raise questions about the initial properties, in which case, revise starting at step 2.
2. **Create property hierarchies.** Determine which properties imply other properties, e.g., in order to carry out an activity, one must be present at the activity. This process may raise questions about the initial properties and classes, in which case, revise starting at step 2.
3. **Property consistency test.** This is the ultimate test of adequacy. Test which and if combinations properties complement each other to answer more complex relevant questions. Test if the specificity of domains and ranges of complementing properties are compatible. (For instance, can some kinds of things have a dimension, but cannot be measured?– if yes, which is the process to determine it?). Find gaps in the reasoning, find modelling patterns and check if variants indicate gaps. According to the application, purely logically derived concepts may be added. Test that **logical consistency** applies throughout the model. Revise from step 2 until here.
4. **Reduce the model.** Remove properties and classes not needed to implement the required functions. Keep them aside for possible future extensions. The smaller the model, the more effective is the information system.

##

## Phase C: Implementation and publishing

The steps are:

1. **Implement the model in a specific knowledge representation language with a specific syntax.** For instance, XML RDF/OWL, TRIG, etc. Verify logical consistency by S/W. According to the application, other purely logically derived concepts may be added.
2. **Write a textual definition.** Write a textual definition with a clear introduction reporting Phase A, all constructs in a syntax independent format that clearly relates to the logic of the model. This may be enriched with FOL statements (Second Order Logic?). Write extensive scope notes to clarify all identity conditions not put in terms of logic.
3. **Install curation.** Install a maintenance team. Identify the authority and authoritative procedure for updates (this can be completely democratic). Provide a public site with a transparent release management and make copies of the implementation and text available under transparent conditions.

## Mapping

 By “mapping” we mean the translation of machine-readable instructions that an automated algorithm can follow in order to transform a set of data organized following a schema A (“source schema”) into a set of data organized following a schema B (“target schema”) ideally without loss or change of meaning. This procedure is used either for migrating data from one format and/or database to another, or for “providing” a copy of some “source” data to a “target” data aggregation service.

In practice, the ideal complete correspondence must be modified by distinguishing the following situations:

1. Only a part of the source schema is of interest for the intended transfer
2. The target schema is more general but less precise than the part of interest of the source schema
3. The target schema misses some constructs to render some meaning of interest in the source schema
4. The target schema contains constructs that overlap with but do not cover a certain source construct.

Case A is trivially met by considering only the part of interest of the source schema respectively. This is characteristic for data provision, because most local systems contain some data not of interest for the target aggregator.

Case B can be described in terms of a “query containment condition”: When data are transformed into the target schema B, for any answer to a query possible under schema A, there should exist a query possible under schema B that returns an answer set that semantically comprises the respective query answer under A.

As illustration, let us consider:

Schema A: Person – attended -> Course, Person – taught -> Course.

Schema B: Actor – participated in -> Meeting.

Mapping: Person -> Actor

 Course -> Meeting

 attended -> participated in

 taught -> participated in.

An instance in A: “George(Person) taught CRM Course(Course)”

 transforms into B: “George(Actor) participated in CRM Course(Meeting)”

An instance in A: “Gerald(Person) attended CRM Course(Course)”

 transforms into B: “Gerald(Actor) participated in CRM Course(Meeting)”

Querying “participated in” in schema B returns:

 “George(Actor) participated in CRM Course(Meeting)”

 “Gerald(Actor) participated in CRM Course(Meeting)”,

 without distinguishing “taught” from “attended” as done in schema A.

The effect of such a mapping is that all facts described by schema A can be found by querying the mapped data under schema B, but the facts may be described less precisely. Not all facts that can be distinguished by querying schema A may be distinguished also by querying schema B. This situation is the ideal case for information integration under a global schema, which cannot be expected to foresee all details of all sources to come. It can be described as a preference of “recall over precision” (see chapters below).

Recognizing such a mapping condition is intellectually the same kind of insight as that in steps 6 and 7 above, only with the difference, that the target classes and properties are already given and not invented or detected in the process. The user must decide, if the given classes and properties of the target contain adequate equivalents or generalizations, or if some additional classes or properties would be needed to be invented or can be taken from other models, not excluding the source itself, in order to cover the ability to map the source.

 Nevertheless, concepts from schema A are missing in schema B. A “trick” helps to overcome this problem, without requiring and extension to schema B: add to classes a “type” and to properties a “role”.

Then,

an instance in A: “George(Person) taught CRM Course(Course)”

 transforms into B: “George(Actor, type: Person) participated in(role: taught)

 CRM Course(Meeting, type: Course)”

An instance in A: “Gerald(Person) attended CRM Course(Course)”

 transforms into B: “Gerald(Actor, type: Person) participated in(role: attended)

 CRM Course(Meeting, type: Course)”

The two generic properties “has type” and “in role” allow for turning source classes “into data”. Then, a loss-free mapping is accomplished. This works as long as roles and types are not related. In schema A2: Student – attended -> Course, Professor – taught -> Course, the mapping to B would still work, but the “binding” of “taught” to “Professor” cannot be rendered by adding types and roles as data. If this is needed on the target side, it would require an extension.

In the following, we distinguish two fundamental use cases: the maintenance of the target schema foresees user-driven extensions, or not. In the latter case, nothing can be done about a true mismatch (case c) above). In case d), overlapping constructs, the best overlapping may be chosen as a surrogate for a true generalization. The mapping job is finished with steps 6 and 7 of the development process above.

If, on the other side, extensions are foreseen (this is the default for using the CRM methodology), the source schema and data encoded under it are seen as empirical source material, and we go back to step 1 of the development process if the source schema has not been sufficiently mapped for the purpose. For any specific construct to be added, and for overlapping concepts (case d) above), suitable generalizations should be considered. In the CRM Methodology we foresee in particular that also existing concepts in the target should be considered to be widened in scope when a particular mapping provides evidence that this is useful for the purpose of the target.

As long as the target schema provides suitable generalizations, the extension is regarded to be “compatible” and can stay in the use of a local community, regardless of if it is proposed as a recommendation for a wider community. If, however, it requires generalizations not covered by the target, it is preferable to modify the target itself. As seen through this lens, mapping is a special case of an ontology development process, especially if the target foresees extension and is open to ongoing modification.

# Principles Introduction

This part of the guide provides a set of individual principles as a reference to be taken into account alongside the Process Model for building and/or extending [ontologies](#_vj2xbzdro28i) for integrating data in [empirical domains](#_jwb1cfff8b8). The principles are organized according to eight thematic areas which each group together a series of related methodological principles and recommendations.

These thematic groups are:

1) Engineering from an empirical base

2) Knowledge structure basics

3) Concept relevance

4) Open world

5) Open world and knowledge progress

6) Open world and knowledge bases

7) Objectivity

8) Language & Concepts

Each of these thematic areas is introduced with regards to its content and is followed by a detailed description of the principles that pertain to the thematic.

At the end of the guide, the principles are reordered in a processual checklist provided to the ontology modeller, which is intended to be used as a reminder and verification tool during the setup, execution, and evaluation of a modelling activity.

That being said, the principles introduced here can also be taken up and applied under a number of different considerations and use scenarios beyond a direct modelling exercise. Particularly they bear a relation to questions of:

**General Ontological Methods (OM)**

Considered in this aspect, the principles enumerated can be applied to the general question of how to understand and then represent reality in formal ontological terms by means of the identification and declaration of formal relations between identifiable classes. The question approached here is: what are the correct means and limits of the representation of reality by a formal ontological construction?

**Ontology Use (OU)**

In implementing an ontological model in an actual [information system](#_rmj4f9tw5j0v), care must be taken in the translation of the ontology as such within a technical environment. Considered in this respect, these principles provide guidance with regards to how to make this translation successfully.

**Knowledge Base/Representation Principles (KB)**

Considered in this regard, the principles intend to inform on the best technical means of representing knowledge and the ability and likelihood to know what are or were stated/known in terms of formal propositions at some time about the world.

**CIDOC CRM Method (CM)**

Considered under this aspect, the principles put forward can provide a guideline to building a formal ontology that can act as a standard. In this regard, the CIDOC CRM in particular is considered and is referenced as an ISO standard which models cultural-historical discourse, including all scientific activities, focusing on a material level of description.

**Principles for Building a Standard (PS)**

In general, we can consider that these principles can inform decisions on establishing manageable [units of documentation](#_y6ql6eualny6). These enable information integration for some domain but also ensure the long-term robustness of the declared classes and relations to support [monotonic](#_r0bgjwaqhz6k) (backward-compatible) revisions of the ontology.

Therefore, each principle given in its main form as a solution can potentially be further elaborated with regards to these various aspects (not carried out in this document).

# Glossary

In this document, we refer to a number of terms which are current now or in the history of discourse on the methodology of ontology development which may not be more or less familiar to the reader. Therefore, we offer here a list of definitions of frequently referenced terms within the document and our interpretation thereof.

**binary relationship**

A relationship between exactly two individual entities, such as one person “is parent of” another person. Some knowledge representation languages, such as RDF, support only binary relationships, so-called “properties”. Any more complex relationship is then described as a class connecting three or more properties, e.g., to describe a temporary membership in a group. Other languages explicitly support relationships between multiple individuals, such as the Entity-Relationship Model.

**class**

A category of items that share one or more common traitsserving as criteria to identify the items belonging to the class. These **properties** need not be explicitly formulated in logical terms, but may be described in a text (here called a **scope note**) that refers to a common conceptualisation of domain experts. The sum of these traits is called the **intension** of the class. A class may be the **domain** or **range** of none, one or more properties formally defined in a model. The formally defined properties need not be part of the intension of their domains or ranges: such properties are optional. An item that belongs to a class is called an **instance** of this class. A class is associated with an open set of real life instances, known as the **extension** of the class. Here “open” is used in the sense that it is generally beyond our capabilities to know all instances of a class in the world and indeed that the future may bring new instances about at any time (**Open World**). Therefore a class cannot be defined by enumerating its instances. A class plays a role analogous to a grammatical noun, and can be completely defined without reference to any other construct (unlike properties**,** which must have an unambiguously defined domain and range). In some contexts, the terms individual class, entity or node are used synonymously with class. A class is a universal.

**closed world**

Describes information systems which assume that the information stored in them is complete relative to the universe of discourse they intend to describe. In particular, statements that cannot be shown to be true are regarded as false.

**complement**

A complement of a class A with respect to one of its superclasses B consists of all instances of its superclass B which are not instances of the class A.

**compression**

A term first used by Fauconnier and Turner (2002)[[3]](#footnote-3) to describe a function of our conscious thinking that reduces complex relationships to seemingly simple ones for particular contexts, without losing the ability to recover the full meaning from the subconscious.

**data**

Encoded information filled into the fields of a data structure for use in a formal way by algorithms and machines, typically managed in an information system in digital format.

**data structure**

Encoded predefined set of typed data fields and rules for their allowed arrangement, which is used as classification and expression of relationships of the data entered into the fields. It is also intended to instruct an information system on how to manage data, typically associated with the fields in a database entry form (the schema of the database).

**concept, primitive**

A class or relation which is not exhaustively defined in terms of logical expressions combining other concepts. For instance: “E21 Person”. It is not a **derived concept**.

**concept, derived**

A class or relation which is exhaustively defined in terms of logical expressions combining other concepts. For instance “mother = female AND Person AND has child”. It is not a **primitive concept**.

**empirical domain**

Domain of scholarship which relies on observations to produce new knowledge. We take this here in a very wide sense, which includes historical documents as observable items and the observations reported in documents, albeit critical about them.

**information**

In the narrower sense meant here: a structured set of symbols compiled by some actor who knows to resolve the meaning of the symbols and intended as a message to other actors supposed to be able to resolve the meaning of the symbols in the same way. It could be as general as texts, but also graphics and in particular data. By resolving the symbols, an actor may turn information into knowledge according to their trust in it.

**information structure**

Any form of grammar needed to interpret the meaning of the relative position of a symbol (or word) in an information unit.

**information source**

Any document or system containing units of information.

**information system**

A computer-based system that allows users to access and communicate information provided by multiple users, without the need of human mediators knowing content, and that allows to reasonably relate information provided by multiple users at different times by means of logical operations in a way useful for their business and in ways not previously obvious to the individual providers.

**information value**

An element or encoded fact in a particular information unit, in contrast to a (reusable) information structure.

**inheritance**

Describes the fact that the properties of a superclass are also properties of its

subclasses.

**instance, class**

An item that can be characterized by a particular class.

**instance, relation**

A factual relationship of a type given by a specific property, between the instance of the domain class and the instance of the range class of this property.

**intension**

The intension of a **class** or **property** is its intended meaning. It consists of one or more common traitsshared by all **instances** of the class or property. These traits need not be explicitly formulated in logical terms, but may just be described in a text (here called a **scope note**) that refers to a conceptualisation common to domain experts. In particular the so-called **primitive** concepts, which make up most of the CRM, cannot be further reduced to other concepts by logical terms.

**knowledge base**

A database that can manage data formatted by a so-called knowledge representation language, such as KL-ONE, TELOS, KIF, DAML, OIL, RDF(S), OWL etc.

**knowledge representation**

Structuring of information by a so-called knowledge representation language, which is supposed to be more similar to the way humans think and the way in which reality is perceived as containing discrete thing than other data encoding paradigms (e.g., E-R, XML).

**monotonicity**

Describes the case in which the relations between classes in an ontology remain valid even if (suitable) new classes and relations are added.

**ontology or ontological model**

Formal naming and definition of the classes and relations of the entities (the stuff) that exist for a domain and can unambiguously be shared in a community of users.

**open world**

Characterizes information systems which manage data consistently with the assumption that the information stored in them is incomplete relative to the universe of discourse they intend to describe. In particular, missing information in the system is not interpreted as non-existence of the respective properties in the universe of discourse.

**particular**

An item that does not have instances. Often an instance of a class.

**perdurant**

Items depending on time. They can only be captured adequately in relation to passing time (e.g. in a video).

**polysemy**

Describes the capacity of a word to have many meanings.

**open world**

Describes information systems which assume that the information stored in them is incomplete relative to the universe of discourse they intend to describe.

**reality**

We regard reality as that which is unique in space and time and makes independent observations potentially comparable about their reference, including mental states of humans.

**relation or property**

A relation defines a link of a specific kind between two classes. The origin class is the domain and the destination class is the range of the relation. A relation is a universal.

**state of affairs**

A representation of an aspect of reality by a set of relationships holding for some time span.

**subclass**

A class that is a specialization of one or more classes (its **superclasses**), i.e. all instances of the subclass are also part of the instances of its superclasses.

**superclass**

A class that is a generalization of one or more classes (its **subclasses**), i.e. all instances of all its subclasses are also part of instances of its superclass.

**tautology**

A proposition that is true in every possible interpretation and therefore cannot lead to new knowledge.

**unit of documentation**

A unit of information managed together due to common provenance and intended communication role, i.e. a file in an IT environment, a book, or an archival entry.

**universal**

An entity that has instances. Classes and properties are universals.

# Engineering from an Empirical Base

The creation of functional integrative ontologies depends on a 'bottom up' strategy of working from real empirical information - data and corresponding [data structures](#_25e6qi7eafg) - in order to abstract relevant [relations](#_xlqxrujni66b) and [classes](#_s5n1e7queg2k) that will adequately cover the modelled domain. By adhering to an evidence-based approach, the conceptual modeller is able to build a model that is capable of providing an explanation/translation of relevant information from target data structures in the domain into a common model. The ultimate criterion for adjudicating such adequacy is the ability of the resultant model to enable scholars/scientists to pose and find answers to their research questions via data described in terms of the proposed model. Therefore, an essential part of the empirical evidence to be gathered includes the high-level research questions that scholars/scientists aim to answer via their data collection. These questions form the necessary contextual basis for understanding and modelling data and conceptualizations.

Under this topic, we identify three principles:

* 1. [Model from existing actually used structured information sources (whenever available)](#_2et92p0)

[1.2 Model according to the research questions justifying the structured information](#_tyjcwt)

[1.3 Model from actual information values](#_3dy6vkm)

## 1.1 Model from existing / actually used structured information sources (whenever available)

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 1.1 | Model from existing / actually used structured information sources (whenever available) | Models should be useful |
| Problem Description |
| What is proper source material for my ontology? |
| Argument / Solution |
| Modeling from existing, actually used structured information (e.g.: databases, spreadsheets, RDF documents, XML documents, structured analogue documents, etc.) ensures that the underlying concepts revealed by the process are useful. It establishes that the model models information that people actually devote resources to encode. Where no existing structured [information sources](#_masqkz62ov4w) exist to begin modelling from, intuitive sketches of potential structured information sources from the target community can be used as an empirical information source. |
| ☺ Eg. | Modeling CRMarcheo from national excavation recording forms |
| ☹ Eg. | Modelling FRBRoo from FRBR, which introduced an intended practice intertwined with existing documentation practice |

|  |
| --- |
| Applicability  |
| OM/CM/PS |

## 1.2 Model according to the research questions justifying the structured information

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 1.2 | Model according to the research questions justifying the structured information | Why do you need this field? |
| Problem Description |
| How can we determine accurately the semantic interpretation needed from potentially ambiguous or overdetermined information sources? |
| Argument / Solution |
| Structured information can have many [senses](#_yho9q24hrmxl). The relation to the actual use of the information can be quite intuitive. The ontological interpretation must follow the real research questions for which the information is used or can be used. This requires the elicitation of (sometimes implicit) research questions from the domain users/community by interview. |
| ☺ Eg. | A field "age" in the CIDOC Relational Model (precursor to CIDOC CRM). This field was fundamentally ambiguous and its use/content could not be understood without reference to interviews with researchers who indicated that it was meant to describe the life phase in which a biological specimen was killed. This interpretation guided the modelling to an unambiguous semantic expression: "Life stage type". Same field age was used by art historians to describe role of artistic artefact in the process of a creating a work. Also: common meaning behind archaeological "find", biological "occurrence", or archaeological "prototype", biological "holotype" |
| ☹ Eg. |  |

|  |
| --- |
| Applicability  |
|  |

## 1.3 Model from actual information values

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 1.3 | Model from actual [information values](#_vdys77uawz7p) | Model only real cases,Model what is meant not what is thought to be meant |
| Problem Description |
| Can we model from the bare [information structure](#_1uyw0o2kz95x) (or do we need to have instantiated information values)? |
| Argument / Solution |
| When modelling the information of interest to represent for researchers, it is important to go beyond the intention of the bare information structures (such as data forms in an information system) to see how they are actually put to use. Information structures such as data forms represent intuitions, local information goals, and practical constraints. They are an intention limited by circumstances. Information actually gathered and entered into structures provides evidence beyond intention, revealing actual practice. Actual practice connects to the real world, reveals exceptions, ambiguities in information structure definitions; it shows not “how it is supposed to be done” but “how it is done”. If in the modelling project in question no information values are yet available, solicit information value samples from researchers. |
| + Eg. | A field "age" in the CIDOC Relational Model was used to describe the products of phases of an artistic process: sketch, underdrawing, etc.Use of a Field Sex: Used both for ‘M/F’ and ‘Yes/No’ values |
| - Eg. | Take "object name" in a collection management system for a proper name |

|  |
| --- |
| Applicability  |
| OM/ CM/PS |

# Knowledge Structure Basics

Knowledge engineering is a practice of creating high-level conceptual models that are capable of explaining structured information occurring within their scope. This practice is aided by an understanding of how the basic elements of a conceptual model's structure (classes and relations) interact as formal units and what their limitations are.

Under this topic, we identify four principles:

[2.1 Detect hidden relations in terms](#_2s8eyo1)

[2.2 Distinguish particulars from universals in the target domain](#_3rdcrjn)

[2.3 Do not define the same property twice for different classes. Find the superclass for it.](#_26in1rg)

[2.4 IsA is an increase of instances and a decrease of properties](#_lnxbz9)

## 2.1 Detect hidden relations in terms

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 2.1 | Detect hidden relations in terms | Who is a creator? |
| Problem Description |
| How does one recognize the essential classes needed for the intended domain from the source material? |
| Argument / Solution |
| Language encourages the projection of relationships into the definition of classes. This projection can occur on the domain or range class. A typical example would the declaration of classes such as "parent" or "child". Here, an implicit relationship is mistaken for two essential classes. In fact, the relationship can be properly modelled as “is parent of” or “is child of” as detected in the domain/world without the need for a declaration of explicit classes for ‘parent’ and ‘child’.  |
| ☺ Eg. | "Parent" as expressed by semantic relation Person "is parent of" Person. |
| ☹ Eg. | TADIRAH "Research Object" |

|  |
| --- |
| Applicability  |
| OM/ CM/PS |

2.1a Detect compressed classes and relations in relations

 use case: 1) go from data structure labels to classes and relations

 2) intuitive conceptual structures

 in both cases, ppl use linguistic structures

 unreflective use of language for the purposes of creating classes and relationships

## 2.2 Distinguish particulars from universals in the target domain

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 2.2 | Distinguish particulars from universals in the target domain | “To be” is not to be“I” am multitudes |
| Problem Description |
| What can I model with a class or relation? |
| Argument / Solution |
| The function of a class or relation is to provide a means to identify real world instances of this class or relation that are referenced in information structures. A class or relation, therefore, operates as a 'universal' in the philosophical sense. That is, it talks about a general category which can have instances. What cannot be modelled with a class or relation is a particular real world thing or relation. Modelling a particular as a class or relation serves no function as it can have no application outside itself. That being said, a universal can have as instance another universal as in classic taxonomy, e.g.: 'species' has instance 'dog' and 'cat’.  |
| ☺ Eg. | **particulars**:  me, “hello”, 2, WW II, the Mona Lisa, thetext on the Rosetta Stone, 2-10-2006, 34N 26E.**universals**: patient, word, number, war, painting, text**“ambiguous” particulars**: numbers, saints, measurement units, geopolitical units.**“strange” universals**: colors, materials, mythological beasts.**Dualisms**: Texts as equivalence classes of documents containing the same text.Classes as objects of discourse, e.g.“chaffinch” and ‘Fringilla coelebs Linnaeus, 1758’ as Linné defined it. |
| ☹ Eg. | Making a place a 'concept' in SKOS. A place is one and particular. It can have no instances only parts. |

|  |
| --- |
| Applicability  |
| OM |

## 2.3 Do not define the same property twice for different classes. Find the superclass for it

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 2.3 | Do not define the same property twice for different classes. Find the superclass for it | No repetition of properties |
| Problem Description |
| What do I do with repeating relations in the modelled data? |
| Argument / Solution |
| The function of an ontology is to provide integration, the essence of which is to find the commonly referred to concepts and relations in a domain. The discovery of repeating properties for different classes suggests that they rely on a common, more general concept, causal to the ability to have such a relation in the first place. Finding the minimal class to describe this common generalization allows the creation of a general class to which the property can be applied and from which this relation can be [inherited](#_j3v0ycwzi129) by assigning the originally modelled classes as [subclasses](#_tlsu33rmr6q7) of the newly created generalization. Creating such classes adds to the efficiency and robustness of the ontological model. This practice often involves coining new terms in order to reference the identified classes/universals that do not have expressions in a particular language natural or formal. |
| ☺ Eg. | "Legal Object" carries the ability to have a right on something, material or immaterial. "Persistent Item" the ability to be present in an event. |
| ☹ Eg. |  |

|  |
| --- |
| Applicability  |
| OM/ CM |

## 2.4 IsA is an increase of instances and a decrease of properties

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 2.4 | IsA is an increase of instances and a decrease of properties  |   |
| Problem Description |
| What is the function of declaring classes as high-level generalizations as opposed to tailored classes to capture specific entity instances? |
| Argument / Solution |
| Modelling an ontology from real information structures and values first suggests lower-level classes and relations that will directly capture instances of the modelled domain. The discovery of common relations amongst these classes will then motivate the declaration of generalized classes. These classes form the domain and range of relations common to the lower-level classes. The result of this process will be an increase of relations moved to higher-level generalized classes which will have more instances and serve an integrating function, while lower-level classes will have fewer relations and less instances but provider greater accuracy. |
| + Eg. | In the CRM ontology, the [perdurants](#_xkvl1b65gk58) branch starts with E2 Temporal Entity which groups together entities of temporal duration and works its ways down to specific classes such as E15 Identifier Assignment. In real world museum practice, museum data is much more likely to become an instance of such low-level classes as E15 Identifier Assignment. The function of the higher-level classes above E15 Identifier Assignment — E13 Attribute Assignment, E7 Activity, etc. — is to gather the relevant higher-level relations which are common to different types of entities of temporal duration. They will have fewer instances, but allow for a more efficient model and higher level of recall. |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

2.5 Candidate (Care has to be taken that if different branches of specializations are made, that they should be tested for harmonization at common class nodes – motivated by 2.3 and/or 4?)

# Concept Relevance

Successful ontology modelling depends on being able to model appropriate and useful relations and classes for the domain under investigation. It should be emphasized that the goal of developing an ontology cannot be to model “everything” but is rather to model the necessary and well-understood concepts and relations for some domain. The principles of conceptual relevance guide the modeller in determining the priorities, limits, and general organization units for the modelling exercise.

Under this topic, we identify four principles:

[3.1 Model primitive concepts first](#_1ksv4uv)

[3.2 A class should allow the formulation of a query that answers a relevant question](#_44sinio)

[3.3 Model manageable units](#_z337ya)

[3.4 Model concepts that express the least interpretational position in order to make the model robust against revision](#_1y810tw)

## 3.1 Model primitive concepts first

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 3.1 | Model [primitive concepts](#_jhkl74g312rj) first |   |
| Problem Description |
| My data suggests a complex list of classes, which should I model first? |
| Argument / Solution |
| [Derived concepts](#_svamowymlz1x) depend on primitive ones. Primitive concepts are those that emerge empirically from constraints of reality along natural gaps, such as the current gap between human and chimp. We cannot understand "professor" if we do not understand "person". If a concept can be determined exhaustively in terms of logical rules in relation to others, it can be computed by a system (and therefore be left unmodelled). That makes integration simpler and avoids redundancy in storage. |
| + Eg. | “Parent” is not primitive, i.e., derived: parent = human & has child. “Human” and “has child” are primitive: only empirically justified. “Parent” as psychological concept is also primitive.  |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OM |

## 3.2 A class should allow the formulation of a query that answers a relevant question

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 3.1 | A class should allow the formulation of a query that answers a relevant question | What question does it answer? |
| Problem Description |
| What is the basic justification for the declaration of a class? |
| Argument / Solution |
| It is insufficient to argue for the declaration of a class based on the fact that it has instances. That concept 'x' is needed to find instances of 'x' is a [tautology](#_yprls57ifyea). A declared class should not only be able to capture relevant instances in the domain in question, but must also be the starting or end point of some relation that is not captured in the [intensional definition](#_2i2kg1cd7w3g) (scope note) of the class and that would appear as a parameter in a relevant query of the modeled domain. |
| ☺ Eg. | Having modelled a general class such as 'car' and given it appropriate relations such as 'has engine type', 'has brand', etc., we have normally then covered our modelling needs for cars. There is no particular motivating factor for creating a subclass for 'subcompacts' or 'porsches' unless our domain of interest makes statements with regarding to all and only these subclasses. Instead, the proposed subclass of model or brand can be put into a type relation, and classification can then be managed through control lists, taxonomies, and thesauri.A second example, sub-types of E55 Type in CRM are typically not needed in order to answer research questions.  |
| ☹ Eg. |  |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

## 3.3 Model manageable units

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 3.3 | Model manageable units | Don't order more than you can eat |
| Problem Description |
| Where does modelling stop? When should I stop adding classes and relations to my model? |
| Argument / Solution |
| There must be a means by which to respond to the recurring request, 'but I need this'. There is no means to model the entire world. The criterion for whether to add classes and relations should be that of integration. Is this information required to be represented by a general ontological model in order to answer common questions of the domain? We can dismiss from the model those things which do not serve the functionality of integration, but must keep in the model aspects that fall within the scope of the ontology. In the case where the scenario is not information integration but information management, the criterion becomes that of functionality and fit for purpose against available resources. This does not mean rare information should not be modelled. |
| + Eg. | The domain of the CRM is historical discourse, ergo the ontology must model/express identifiers, documents, and types (as used objects). Not in the scope of basic CRM would be local administrative actions, specific conservation routines, etc. Therefore, there things are not represented by classes or relations in CRMbasic. This does not mean that they cannot be modelled, but their modelling would fall to an extension with a scope that should be well defined and justified. |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OM/ CM/PS |

## 3.4 Model concepts that express the least interpretational position in order to make the model robust against revision

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 3.4 | Model concepts that express the least interpretational position in order to make the model robust against revision  | How to make a core model |
| Problem Description |
| What classes in general should receive priority in modelling? |
| Argument / Solution |
| In order to ensure that a model can support monotonic revision (no need for basic reclassification), classes that can be generally accepted by the target community should be given modelling priority. These are the stable points of discourse from which generalizations or specializations can be made. Their instances are also the stable points of discourse along which diverging opinions about the nature of things can be integrated. Classes describing instances that are more controversial/difficult to be verified can be added under these more robust classes as and when sufficient evidence is gathered to support a stable declaration. |
| + Eg. | ‘States’ are not modelled in CRMbasic because they are subject to strong interpretational ambiguity (and therefore false instance association).  |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OM/ CM/PS |

# Open World

The principle of [open world](#_e2e3umrx2zse) deeply affects how we undertake conceptual modelling tasks. It is paramount that we take this principle into account when declaring classes and relations in order to be faithful to this basic condition.

For the purposes of conceptual engineering and information systems development, we must adopt the open world principle: at the level of the model and at the level of the data, and with regards to the management of the knowledge base. On the level of the model, we can only know that we have not modelled the whole world, not even a closed part of it, as long as it is not completely controlled by predefined rules and sufficient observation. The model depends on modelling real world epistemic processes, and these are themselves inherently open. Therefore, modelling must take place under the constraint of open world at all times. At the level of data again we cannot impose closed world constraints because of the incompleteness of our particular knowledge at any one time.

Under this topic, we identify three principles:

[4.1 Never define a class as complement](#_2xcytpi)

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 4.1 | Model a concept by defining what it is, not what it is not | Non-elephants are not a kind |
| Problem Description |
| Describing a set of sibling subclasses, it is tempting to make a complete set that covers all possible meanings of the superclass, in particular each one excluding the others. If not all meanings can be described, one is tempted to make one class for all “other things” not already described by the other siblings, i.e., the logical complement of the other sibling classes.  |
| Argument / Solution |
| In an Open World, the exclusion of known things cannot provide an identifiable substance, because it cannot be foreseen which phenomena may exist or will occur. Therefore, no property can reliably be associated with a complement. Further, any future encounter of things violating the distinction will need a non-monotonic revision. A complete partitioning into sibling subclasses is a variant of the complement: it means that the complement is empty.The solution is not to define the complement, but instead to instantiate the superclass only with its instances, typically using the property *P2 has type* to refine the particular meaning of such instances. |
| + Eg. | E37 Mark is defined as superclass of E34 Inscription, i.e., as everything inscribed, including all non-text marks, in order to have a generalized location for items when it cannot be determined whether they are inscriptions in the narrower sense or not. All instances of E37 Mark that are not instances of E34 Inscriptions can still be characterized by a suitable terminology using *P2 has type*. |
| - Eg. | Do not define subclasses of E21 Person based on sex or gender. Do not define E37 Mark explicitly as not text and only symbols, in contrast to instances of E34 Inscription.  |

## 4.1 Never define a class as complement

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 4.1 | Never define a class as a [complement](#_hfbbfazabudz) | Open number of siblings! Caution with disjoint classes! No non-elephants |
| Problem Description |
| If a class has one or more subclasses, where do we put instances that are not described by either of these? |
| Argument / Solution |
| A complement class cannot have any property of its own other than just being "not" another thing. If it did, we would know everything about this "not" being, which violates the Open World assumption. Such a negative class declaration would entail that we know all possible subclasses that could occur under some superclass, excluding all and any new possibilities. If it did not, then this negative class would have no substance of its own and therefore say nothing of value. Therefore, the recommendation is to make instances of the unknown sibling subclasses instances of the next superclass. This entails that the model should have no "abstract classes" in the sense of having no direct instances. Any instance has more properties than any class which means that we can always find additional more specific classes. |
| ☺ Eg. | Declaration of complements like M/F easily falsifiable. "Physical Feature" is not the complement of "Physical Object". What about buildings? |
| ☹ Eg. | Non-information objects in Europeana |

|  |
| --- |
| Applicability  |
| OM / OU / CM / PS |

## 4.2 Cover incomplete details of knowledge by what you do know

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 4.2 | Cover incomplete details of knowledge by what you do know | There's always more detail to it |
| Problem Description |
| How can I represent characteristic states of lack of knowledge in my modelling? |
| Argument / Solution |
| Model should both support the more complete, accurate picture of a potentially complete state of knowledge but also provide shortcuts that allow the representation of the domain's characteristic states of lack of knowledge.  |
| ☺ Eg. | Dimension: Often a dimension is recorded as, for example, “10cm wide”, but by what measure, done when, by whom? In the CRM the concept of dimension has a full modelling which is the product of a measurement activity (E16 Measurement) and allows for a complete documentation of the actual state of affairs. The typical knowledge, however, does not include all these details. Therefore, we include also a direct shortcut to E54 Dimension to represent typical knowledge situations, while leaving open the possibility of enrichment at a later date if possible. ... Define P53 has former or current location (is former or current location of) as any wider area.B) The lamp hangs on a links on a link ……..on a link on the ceiling. ... C) Define P53 has former or current location (is former or current location of) as any wider area in which something is. |
| ☹ Eg. |  |

|  |
| --- |
| Applicability  |
| OM/CM/PS |

## 4.3 Do not create closed worlds of properties

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 4.3 | Do not create [closed worlds](#_ei3j7beiwq5m) of properties | Leave the door ajar |
| Problem Description |
| Should I create rules to indicate the strict logical possibilities of relations amongst my classes? |
| Argument / Solution |
| Using the open world assumption, we cannot model closed worlds of properties. What is detailed in the model is the world that we have derived within a limited scope according to evidence of certain data structures. New input may add additional means of relating classes that we have not yet foreseen. Therefore, it would be an error to do so. |
| + Eg. | An E85 Joining event cannot be inferred from an instance of P107 has current or former member (is current or former member of) (decision by CRM-SIG) “you can be member of a group only by joining”: Someone may become member of a group by birth, and by what else? (Discussion in CRM-SIG) |
| - Eg. | To declare that a class that has only a limited set of properties with no potential expansion.Information Carrier: class made to carry information OR thing that carries information OR thing that may carry information.Painters paint paintings. |

|  |
| --- |
| Applicability  |
| OM/OU/KB/CM/PS |

# Open World and Knowledge Progress

The principle of open world affects not only how we should immediately model but also how to manage the development of the model. The open world assumption affects choices on how to build a model in such a way that it can handle knowledge progress/revision from new facts discovered in the data of the domain.

Under this topic, we identify four principles:

[5.1 Support progressive improvement of classification knowledge by IsA hierarchy](#_qsh70q)

[5.2 Do not model conclusions before and without their reasons](#_3as4poj)

[5.3 Describe the intension of and declare classes that model the parts of the domain you understand](#_1pxezwc)

[5.4 Model domains and range or properties consistent with your level of knowledge of the domain of discourse](#_49x2ik5)

## 5.1 Support progressive improvement of classification knowledge by IsA hierarchy

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 5.1 | Support progressive improvement of classification knowledge by IsA hierarchy | Even if you don't know the particular, you may know something more general |
| Problem Description |
| How to support progress of knowledge without completely invalidating old results?  |
| Argument / Solution |
| Use of IsA hierarchy, which has consistency of substance in its definition, allows the representation of different levels of knowledge. The more general levels should model what typically is more likely to be known in the domain of discourse in the absence of more precise knowledge of some instance.  |
| + Eg. | Example of author: if not known person or institution, actorExample of participation: if not known role within an action, ‘participated’, this can be specialized and refined if we gain new information. (Rashomon) |
| - Eg. | Carmine in AAT used generalization “used as dye and pigment”.  |

|  |
| --- |
| Applicability  |
| OM/OU/KB/CM/PS |

## 5.2 Do not model conclusions before and without their reasons

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 5.2 | Do not model conclusions before and without their reasons |  |
| Problem Description |
| How can I try to ensure a class will support multiple knowledge revisions at data level without itself having to be revised?  |
| Argument / Solution |
| Integration relies on data modelled under the system not needing to be fundamentally revised on the input of new data, because different sources may know different parts of the same world, some possibly giving more details than others. Be robust against increase of knowledge. Monotonicity of primary knowledge: new facts not in contradiction with previous ones should not invalidate the representation of the previous. This can be done by making sure to model classes that represent the generally acceptable [state-of-affairs](#_e6liygs69xnq), not a particular interpretation of how those states of affairs came about. |
| + Eg. | Oetzi knowledge revision... facts remain the same, but more details added or noticed. This radically changes the interpretation, but facts remain stable. Death is a robust event class here. Interpretation attempts to assign more causality based on new evidence. |
| - Eg. | State: because is very difficult to know that a state of affairs actually existed in its entirety from start to finish, states are bad classes to declare in a model aiming for integration. |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

## 5.3 Describe the intension of and declare classes that model the parts of the domain you understand

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 5.3 | Declare and describe the intension of classes that model the parts of the domain you understand |  |
| Problem Description |
| How can I handle extension of my model in a way that is open to revision and addition without causing problems of data revision on model update? |
| Argument / Solution |
| If initial classes and relations model just what we know about a domain, and generalizations of these classes are fit to purpose for generalizing just over this domain, then we have a stable basis to extend the model indefinitely. Extensions can either be introduced as specializations or generalizations on the existing model, thus preserving monotonic revision. |
| + Eg. | Defining E18 Physical Thing as highest form of material things (stability of form), even though blood samples are not covered. This supports a "part-of" concept. Later you can add "Material Substantial". Never define a class as complement |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OM/ CM/PS |

## 5.4 Model domains and range or properties consistent with your level of knowledge of the domain of discourse

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 5.4 | Model properties or domains/ranges in a manner consistent with your level of knowledge from domain of discourse  |  |
| Problem Description |
| The relation I model could cover a very wide array of instances, how far should I leave it open or restrict it? |
| Argument / Solution |
| Although a relation may indicate a possible wide range even beyond one’s modelled world, restrict it to what is known from the domain. When an instance exhibiting this property is encountered that falls out of the current domain and/or range take your world as having been a restriction of a new, wider one, and increase the domain and/or range (which is backwards compatible). Hence, even though the property implies its maximal domain and range, we do the opposite in practice and model its minima, to be safe. |
| + Eg. | "Actor" is restricted to human beings and their groups as being able to "perform" activities. You can later add New Caledonean Crows, Dolphins, Chimps and Keas. Similar: parts of Physical Things |
| - Eg. | E1 Entity has dimension. E81TransformationP123 resulted in (resulted from): E77 Persistent Item. E70 Thing P43 has dimension (is dimension of): E54 Dimension |

|  |
| --- |
| Applicability  |
| OM / CM/ PS |

# Open World and Knowledge Base

The same world states may be described in different knowledge bases by different selections of facts according to the processes and available knowledge of their maintainers. About particular states-of-affairs, alternative opinions may be held without obvious ways to consolidate them at the current state of knowledge.

Under this topic, we identify four principles:

[6.1 The absence of a property in the knowledge base is not its negation in reality](#_147n2zr)

[6.2 Allow alternatives or contradictions in the data](#_3o7alnk)

[6.3 Make sure alternative assertions can be unambiguously related to a single entity](#_23ckvvd)

[6.4 Explain Data Structures](#_ihv636)

## 6.1 The absence of a property in the knowledge base is not its negation in reality

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 6.1 | The absence of a property in the knowledge base is not its negation in reality  |  |
| Problem Description |
| How can we interpret/use absence of a property in the knowledge base? |
| Argument / Solution |
| We cannot impose an ontological structure of the actual world in the knowledge base. The model itself gives the real possible relations of the world as ontological structure. Data encoded in the model and stored in the knowledge base, however, relates to our state-of-knowledge. Our state-of-knowledge may be incomplete with regards to the facts. Therefore, neither is it a requirement to use a property from the model nor does its lack of instantiation indicate its lack of existence for an instance. |
| + Eg. | The case of the father: every person can be said to have one biological father, and we can model this in our ontology. But in our knowledge base, we may not have the information required to encode who is the father. The knowledge base must not be expected to hold information that we do not have about what is the case. (It is not the case e.g., not knowing a father means not not having one) |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OU/KB |

## 6.2 Allow alternatives or contradictions in the data

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 6.2 | Allow alternatives or contradictions in the data | Let 100 flowers blossom |
| Problem Description |
| How can we faithfully represent that our state of knowledge with regards to some states-of-affairs may not admit or allow for a single conclusion at some point in time? |
| Argument / Solution |
| What is the case in the world may allow only one true right answer; however, our state-of-knowledge may not allow us to say what is the case but only to give the possible versions of the case. To adequately represent the available knowledge, we must be able to represent its indeterminate or plural state. Therefore, the knowledge base should admit multiple, potentially contradictory statements with regards to the same state of affairs. Contradiction is to be supported at the level of the knowledge base, not the model. |
| + Eg. | Multiple fathers case: there can only have been one biological father. But we do not know which.  |
| - Eg. |  |

|  |
| --- |
| Applicability  |
| OU / KB |

## 6.3 Make sure alternative assertions can be unambiguously related to a single entity

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 6.3 | Make sure alternative assertions can be unambiguously related to a single entity |  |
| Problem Description |
|  |
| Argument / Solution |
| The model should provide an unambiguous class at which to find alternative or contradictory assertions about a particular individual. |
| + Eg. | Alternative assertions about artist behind a particular painting to be found at the creation event. |
| - Eg. | Find the creator associated to different paintings directly or in biography. |

|  |
| --- |
| Applicability  |
|  |

## 6.4 Explain Data Structures

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 6.4 | Explain data structures | Explain, don’t prescribe |
| Problem Description |
| What is the role of an ontology as a standard in implementing a knowledge base? Should it explain data structures or dictate them? |
| Argument / Solution |
| To meet the integration goal for the purposes of epistemic processes, an ontology must be explanatory not prescriptive. It is derived from the world, and meaning does not depend on accidental knowing. Therefore, completeness of knowledge cannot be enforced at integration time. It depends on very specific context, if certain information can be ensured to exist. An explanatory ontology can also be used to motivate better data structures on a technical level. |
| + Eg. | No CRM property is ‘mandatory’ |
| - Eg. | Getty’s ‘Object ID’, the EAD |

|  |
| --- |
| Applicability  |
|  |

# Objectivity

The principle of objectivity is key in building ontologies that can serve the function of integration since it ensures that modelled information can be identified and retrieved by independent users regardless of contextual background. It imposes a standard of clarity and impartiality that allows only data that can potentially be assessed independently by third parties to be modelled and incorporated. This also has a number of positive results in terms of efficiency of the model.

Under this topic, we identify four principles:

[7.1 Be view neutral](#_41mghml)

[7.2 Avoid concepts depending on a personal/ spectator perspective](#_2grqrue)

[7.3 Avoid concepts depending on accidental and uncontextual properties](#_vx1227)

[7.4 Maintain independence from scale](#_3fwokq0)

## 7.1 Be view neutral

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 7.1 | Be view-neutral | Take the middle ground |
| Problem Description |
| How should one represent concepts that can be described differently depending on the observer/documentalist’s relative position within the situation? |
| Argument / Solution |
| Reduce complexity by declaring view independent/neutral classes and relations. This makes a simpler model and allows reference to same thing by parties taking different positions. |
| + Eg. | Transaction, Acquisition, Transfer |
| - Eg. | Buying, Selling, Delivering, Receiving. "Object Name is Pencil in a museum that has only one pencil." |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

## 7.2 Avoid concepts depending on a personal/ spectator perspective

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 7.2 | Avoid concepts depending on a personal/spectator perspective   |  |
| Problem Description |
| How can we make sure other users will be able to identify things in the same way as instances of common classes or relations? |
| Argument / Solution |
| For the purposes of integration, an ontological model must express facts that are verifiable and reidentifiable by objective criteria. The observer's subjective view cannot be reverified by another actor, and as such represents an epistemic state rather than a referenceable objective entity (except qua state itself). |
| ☺ Eg. | Describing people in my photograph as links of type ‘represents’ to my photo (not a group of people).  |
| ☹ Eg. | e.g., “The group of people in my photo” => epistemological units, “orthogonal” to the ontology as a “theory of being”.Difference between curated physical holding and a list of references. |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

## 7.3 Avoid concepts depending on accidental and uncontextual properties

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 7.3 | Avoid concepts depending on accidental and uncontextual properties  |  |
| Problem Description |
| How can one model objectively recoverable entities, available to verification by users across the domain? |
| Argument / Solution |
| Different institutional, disciplinary, or personal perspectives may merge accidental properties into the definition of a class. Such classes are not functional for serving as a definition for an objectively recoverable set of objects. Such class definitions presuppose and entail some context which is left unexpressed. Strip declared classes of such presuppositions in order to model the objective referent without its unexpressed content. Such features can be modelled separately once rendered explicit. Find the actual substance once an accidental relation is removed, if there is any, of the modelled class.  |
| + Eg. |  |
| - Eg. | "Research Object" (TADIRAH), "Aggregated Resource (ORE Model)", “museum object”, “Buhmann (bogey-man)”. |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

## 7.4 Maintain independence from scale

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 7.4 | Maintain independence from scale | The biggest dwarf is larger than the smallest giant |
| Problem Description |
| How should one model different relative levels of size of entities? |
| Argument / Solution |
| There are no objective categorical boundaries between things of different size. If things exhibit characteristic sizes, they are due to other, substantial factors that should be modeled in the first place. For instance, mammals exhibit characteristic limits of size (whale versus shrew) due to metabolism constraints, but in between there are no distinct sizes. Size is a quantitative property. |
| ☺ Eg. | introduce a scale-independent superclass: "settlement"  |
| ☹ Eg. | “hamlet – village”, "ship-boat" |

|  |
| --- |
| Applicability  |
| OM /CM/PS |

# Language and Concepts

Conceptual modelling must distinguish and track between the terms used in a domain and in natural language to express propositions and the conceptual structure that lies behind these expressions. Keeping a clear distinction between linguistic and conceptual levels delivers a number of important principles to bear in mind during modelling activities.

8.1 Don’t confuse polysemy with multiple abstractions

8.2 Detect temporal entities behind relationships

8.3 Don’t confuse class with a term from natural language (proposed)

##

##

## 8.1 Don’t confuse polysemy with multiple abstractions

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 8.1 | Don’t confuse [polysemy](#_zfuvnnegbqdu) with multiple abstractions |  |
| Problem Description |
| How can we respond to the problem of polysemy? |
| Argument / Solution |
| Polysemy represents a challenge to correct modelling. To meet this problem, we must clearly distinguish terms from concepts. A polysemic term does not entail that all meanings belong to the same abstraction. When dealing with polysemy we must not model the relation of terms to terms, but the multiple referents suggested by the polysemy and the nature of these referred objects. |
| ☺ Eg. | Polysemy: Can a museum take decisions? Can I walk into the museum? Can I move the museum?Modeling museum as organization, building, collection respectively<ultiple abstraction: Person IsA Actor, Physical Object |
| ☹ Eg. | Declaring one class for museum in all its senses |

|  |
| --- |
| Applicability  |
| OM |

## 8.2 Most binary relationships in intuitive conceptualizations conceal temporal entities

|  |  |  |
| --- | --- | --- |
| ID | Principle | Slogan |
| 8.2 | Most [binary relationships](#_gpdyjvr7d35m) acquire substance as temporal entities | When did the author write it? |
| Problem Description |
| Should we model directly after a simple phrase model of the relations we wish to represent? |
| Argument / Solution |
| Regular phrases that might translate a data structure often contain [compressions](#_6e4b72z4ofx4) which need to be made explicit and modelled. In order to elicit these compressions, it is useful to think of additional queries that one would want to make relevant to the target phrase and see if it helps make explicit the implicit concepts and relations. This principle is especially important in order to understand hidden events in data structures. |
| + Eg. | Birth (allows connection of child, mother, father through one node), Production (allows connection of actors, tools etc. through one node) |
| - Eg. | “has met”, “has created”, “was added to” |

|  |
| --- |
| Applicability  |
| OM |

# Conceptual Modelling Checklist

Preflight

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Have I gathered original data sources from which to model? | 1.1 |  |  |
| Have I checked the actual data modelled in data forms and not just the data structures? | 1.3 |  |  |
| Have I gathered relevant research questions? | 1.2 |  |  |

Initial Modelling

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Have I separated tried to detect hidden relations by critiquing classes in my sources? | 2.1 |  |  |
| Are all my classes universals (only universals have instances)? | 2.2 |  |  |
| Have I eliminated all semantically duplicate properties by declaring appropriate super classes?  | 2.3 |  |  |
| Are all my classes primitives (indicating a natural, identifiable fold in reality) or derivatives of primitives (where I have a good reason to model them)? | 3.3 |  |  |
| Have I modelled relations to cover the most well-known and secure cases of its use in my domain?  | 5.3 |  |  |
| Have I modelled only concepts as opposed to terms? | 8.1 |  |  |
| Do my relationships sufficiently parse hidden events out of natural language phrases? | 8.2 |  |  |

Model Relevance Check

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Do all my classes allow me to answer a question and have at least one property? | 3.1 |  |  |
| Do my classes and properties have relevance across the domain or do they model local data with no global significance? | 3.2 |  |  |

Open World Compatibility Check

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Have I deleted any complement classes? | 4.1 |  |  |
| Have I introduced suitable shortcuts for characteristic states of lack of knowledge? | 4.2 |  |  |
| Have I left logic of relations open so as to allow new, previously unconsidered fact types to be expressed? | 4.3 |  |  |

Model Objectivity Check

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Are my declared classes view neutral? | 7.1 |  |  |
| Have I purged any class reliant on a subjective view? | 7.2 |  |  |
| Do my classes represent entities that can be re-identified without a given institutional/disciplinary perspective? | 7.3 |  |  |

Knowledge Base Check

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Does my KB allow incomplete information in the data? | 6.3 |  |  |
| Des my KB support contradiction in the data? | 6.2 |  |  |
| Does the IsA hierarchy in my KB allow the expression of different states of knowledge? | 5.1 |  |  |

Model Revision Robustness Check

|  |  |  |  |
| --- | --- | --- | --- |
| Check  | P | Yes/No | Comment |
| Does my suggested new class or property generalize or specialize an existing class or property? Is it consistent with the substance of the generalized/specialized class/relations? | 5.3 |  |  |
| Are my classes/relations constructed so as to support progressive reasoning process by providing suitable neutral, factual abstractions? | 5.4 |  |  |

1. For instance, weight of a living being is in constant flux. Length of wood changes with humidity. A coast line is both a fractal penetration of water and land and dependent on waves and sea level. A car crash begins between a driver’s mistake and touching metal. [↑](#footnote-ref-1)
2. Aldo Gangemi 2006 [↑](#footnote-ref-2)
3. FAUCONNIER, G., AND TURNER, M., 2002. *The Way we Think : Conceptual Blending and the Mind’s Complexities*, Basic Books, New York. [↑](#footnote-ref-3)