Ontology Representation : design patterns and ontologies that make sense

Hoekstra, R.J.

Publication date
2009

Chapter 5

Ontology Engineering

> A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.
>
> *Douglas Adams*

5.1 Introduction

Over the years several design principles have been cast in ontology engineering methodologies to improve the quality of ontology development. And fortunately, ontologies are not necessarily drafted from scratch. The widespread availability of ontologies on the web has opened the door to the adoption of pre-existing general ontologies. By providing an initial structure and a set of basic concepts and relations, these ontologies can be a valuable jump start for more specific ontology development.

On the other hand, an ontology is not just *any* terminological knowledge base, but rather embodies a specific definitional perspective. Where the representation of terminological knowledge is based on Minsky's notion of a frame – concepts are defined by the typical context in which they occur – an ontology refines this notion and needs to distinguish between the inherent and accidental properties of concepts (Breuker and Hoekstra, 2004c; Guarino and Welty, 2002; Bodenreider et al., 2004). Although a typical property such as 'position' is clearly relevant from a knowledge engineering perspective, ontologically speaking it is usually a side issue: changing the position of an object does not make it intrinsically different. This *epistemological promiscuity* of terminological knowledge representation has led to the formulation of several design principles, such as in ONTOCLEAN (Guarino and Welty, 2002, 2004) and in Breuker and Hoekstra (2004c); Breuker et al. (2004); Hoekstra et al. (2008).

A more recent development is the identification of *design patterns* that are meant to overcome some of the limitations in the reuse of existing ontologies and the application of design principles. Where the former are frequently quite large and heavyweight, and are therefore hard to mould and extend to a usable domain ontology, the latter are quite abstract and difficult to translate into concrete definitions in the ontology. Design patterns offer a middle ground between the two, where the methodological principles are made concrete in manageable 'bite-sized' chunks (Gangemi, 2005, a.o.). Furthermore, these patterns can assist in tackling problems related to the ontology representation language of choice (Hoekstra and Breuker, 2008, a.o.).

This chapter gives an overview of each of these four approaches and discusses their strong points and weaknesses.

5.2 Criteria and Methodology

During the mid-nineties, experience in several large scale ontologies led to a widespread consensus that the ontology engineering field was in need of methodological grounding. Independently, the design principles and experience of several ontology efforts were put to paper. Most influential, in this respect are the experiences of the TOVE (Grüninger and Fox, 1995, TOronto Virtual Enterprise) and Enterprise ontologies (Uschold and King, 1995). Both ontologies were developed to enable the specification and sharing of descriptions of business processes. On the basis of experiences in ontology development for the Ontolingua system in Gruber (1993), Gruber identified a trade off between five design criteria (Gruber, 1994):

Clarity
An ontology should effectively communicate the intended meaning of defined terms. Definitions should be objective, formal (if possible), and documented with natural language.

Coherence
An ontology should sanction inferences that are consistent with the definitions.

Extensibility
An ontology should be structured such that it can be extended and specialised monotonically, i.e. without needing to change the ontology itself.

Minimal encoding bias
The conceptualisation should be specified at the knowledge level, without depending on a particular symbol level encoding, e.g. constructions for convenience of notation or implementation.

Minimal ontological commitment
An ontology should only enforce the minimal commitment necessary to support the intended knowledge sharing activities.

The trade-off lies in the fact that these criteria cannot be considered independently. The clarity and coherence of an ontology benefits from a formal specification. But every formal specification involves an inherent concession

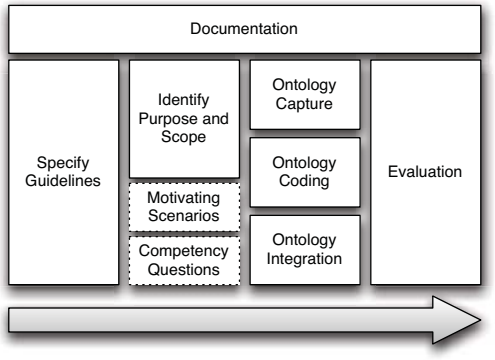

Figure 5.1: Phases in the methodology of Uschold and Grüninger (1996)

with respect to the encoding bias of the language used. With the arrival of the Semantic Web and OWL, most of these parameters have to some extent been fixed (see Chapter 3).

Although use of OWL as ontology representation language involves a significant commitment to a particular symbol level encoding, the encoding itself is designed for maximal clarity and coherence. Extensibility is ensured through its open world assumption and imports mechanism, and since OWL is currently *the* standard for sharing ontologies across the web, it is furthermore likely that a commitment to this language facilitates rather than impedes sharing. Nonetheless these design criteria are not automatically upheld once we adopt the built-in restrictions of OWL: there is no silver bullet. The language leaves a substantial amount of legroom at the conceptual level.

In Uschold (1996); Uschold and Grüninger (1996) experiences in developing the TOVE and Enterprise ontologies were combined in a skeletal methodology for ontology development that incorporates the principles set out by Gruber (1994). Rather than as a precise step-by-step instruction for ontology development, the methodology of Uschold (1996); Uschold and Grüninger (1996) is illustrated in terms of its main characteristics (see Figure 5.1). First of all, they advocate the description of a clear set of guidelines that govern the development process as a whole. These guidelines may range from relatively straightforward agreements, such as naming conventions, to the basic principles of Gruber, and even to formal editorial voting procedures. A clear example of the latter can be found in the OBO Foundry (Barry Smith, 2007); a coordinated effort to develop and integrate multiple ontologies in the biomedical domain.[1] Changes to one of the ontologies in the foundry need to be accepted by an editorial board, and possible overlaps between ontologies are solved via arbitration. One of the points stressed by Uschold and Grüninger is to have guidelines in place for the development of *documentation* during development. It can be quite hard to reconstruct the reasons underlying a particular modelling decision, and having documentation in place at an early stage increases the chance that similar problems are solved in a similar fashion.

[1]The Open Biomedical Ontologies Foundry (OBO Foundry) is freely accessible online, see `http://obofoundry.org`.

In the second phase, purpose and scope of the ontology are identified. Having a clear understanding of why the ontology is developed, its possible use and users, gives a handle on the design decisions necessary in consecutive phases. Grüninger and Fox (1995) advocate the description of motivating scenarios that explain possible use of the ontology and use these to formulate a set of competency questions that should be answerable given the terminology to be defined in the ontology. The purpose of an ontology should not be conflated with the task-dependence of knowledge based system development. In principle an ontology does not sanction a particular use of its contents, but it may be designed to fit a purpose *as a whole*, i.e. how the ontology is used in relation to other knowledge components. The scope of an ontology is its maximal potential coverage in a domain, that is all knowledge inferable from its axioms. Scope therefore only partly relates to the size of the ontology.

Ontology building is the next phase and consists of an interplay between three closely related tasks: capture, coding and integration. Ontology capture is the identification of key concepts and relations, the production of precise definitions in natural language, and the important task of choosing the proper term to refer to these ontology elements. Natural language definitions are in fact the primary source of documentation for people unacquainted with the resulting ontology. Ontology browsers such as Ontolingua and the more recent TONES Repository browser,[2] as well as the mainstream ontology editors Protégé and TopBraid present axioms and local annotations in the same view.

Once the main concepts and relations have been identified and described, they are to be represented in some language. Ontology coding thus naturally involves a commitment to a particular knowledge representation formalism; a commitment that needs to be well motivated because the level of formality has a direct effect on the diligence required of the ontology engineer. The choice for a particular knowledge representation language involves two important commitments (see Chapter 4):

- A *direct* ontological commitment to the structure imposed on the world by the primitives and semantics of that language, and secondly,
- a *more formal* definition of a concept implies a *stronger commitment* to a particular interpretation.

The most suitable level of formality is determined by the purpose of the ontology. A knowledge representation ontology that is to play a part in reasoning requires a more limited formal language than a formal ontology in philosophy, see Section 4.2. Similarly, a knowledge management ontology that is merely used to annotate resources for the purposes of lightweight information retrieval inspires less concern for logical consequence, but still calls for machine interpretability. The developer of an ontology that merely expresses an agreed upon vocabulary to be used within a community of practice might even skip the coding step entirely and suffice with a list of definitions in natural language only. For example, it may not be very useful for languages such as the Dublin Core metadata initiative[3], FOAF[4] and RSS[5] to include exact ontologic-

[2]See `http://owl.cs.manchester.ac.uk/repository/browser`

[3]See `http://dublincore.org/documents/dces/`

[4]Friend Of A Friend, see `http://www.foaf-project.org/`

[5]RDF Site Summary, see `http://en.wikipedia.org/wiki/RSS_(file_format)`. RSS currently has various dialects.

ally concise, rigourous formal definition of the exact meaning of concepts such as 'author', 'friend' or 'news'. On the other hand, a formally specified biomedical ontology may be used in conjunction with a standard reasoning engine to automatically classify proteins into functional classes (cf. Section 7.4).

The third task in the building phase is to take advantage of knowledge reuse by integrating the ontology with already existing ontologies. A natural step, it may seem, but in practice this can be quite demanding. First of all, it requires a thorough understanding of the imported ontology. This is because importing another ontology does put a strain on the principle of minimal ontological commitment: it may significantly extend the size of your ontology. The mere fact that some other ontology contains a proper definition of some concept that is within scope, is often not enough reason to include it. Are the *other* concepts in the imported ontology within scope, and does the imported ontology support our purpose? It should be clear what the assumptions underlying the other ontology are. Secondly, if formally specified, the imported ontology must at least be formally compatible with the knowledge representation formalism selected in the coding task. Arguably, as Uschold and Grüninger (1996) point out, the three tasks should not be performed in sequence. The choice for a particular formalism in part depends on the availability of suitably specified other ontologies.

In the final phase the ontology is evaluated with respect to the requirements specified earlier. If a formal language is used, the competency questions can be reformulated in this language to check compliance and coverage. Secondly, if deviations from the initial outset become clear, it should be checked whether these are well motivated.

5.2.1 A Social Activity?

Most methodologies followed in the footsteps of earlier experiences in knowledge acquisition, such as the CommonKADS approach of Schreiber et al. (2000). The methodologies of Uschold and King (1995); Fernández et al. (1997) invariably adopt a typical knowledge acquisition approach aimed at maximising coverage. Expert interviews, brainstorming sessions and text analysis are used to gather as much information about the domain as possible. This information is then used as the resource material for the actual construction of the ontology.

However, inspiration to reach a mature ontology engineering methodology was sought in adjacent engineering fields in computer science as well. The METHONTOLOGY methodology of Fernández et al. (1997); Fernández-López and Gómez-Pérez (2002) incorporated best practices from software engineering methodologies. In particular, Fernández-López and Gómez-Pérez (2002) regard ontology engineering as a software development process, and introduces the notion of an ontology development *lifecycle*: rather than building ontologies from scratch, it emphasises development as a continuous refactoring of already existing ontologies.

When Gruber formulated his criteria, the most prominent use case for ontologies was the ability to share and reuse knowledge across both systems and communities. The prevalent expectation was therefore that ontologies would be *large scale* and built by *many different people* in concert. Since ontologies are intended to capture a shared view on some domain, their development is much

akin to a process of *standardisation*. According to this view, ontology specification is about reaching consensus on the meaning of terms; it is a *social activity*.

Considering some of the most prominent ontologies we see today, this expectation is only partially fulfilled. A first observation is that ontologies vary widely on the scale of the standardisation involved. On the one hand we *can* distinguish ontologies that are the result of large scale standardisation efforts. On the other hand, most ontologies currently available are relatively small, or the result of an academic exercise of only a few people. Secondly, formal ontologies used and developed by larger communities appear to exist only for scientific domains. The obvious examples that come to mind here are the efforts in the life sciences mentioned earlier, such as SNOMED, the Gene Ontology and GALEN. This is not very surprising as (like ontology) science is all about primitives; about discerning entities in reality by sometimes literally splitting them up into smaller bits. Both Mendeleev's periodic table of chemical elements and the Linnaean taxonomy of species are some of the earliest ontologies and were developed to capture, or rather facilitate the expression of a scientific theory.

It is hard to find a large scale ontology that *is* the result of a community effort. General ontologies, such as the Basic Formal Ontology (BFO, Grenon (2003)), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE, Masolo et al. (2003)), Sowa's top ontology (Sowa, 2000) and the LKIF Core Ontology of Basic Legal Concepts (Breuker et al., 2007; Hoekstra et al., 2007, this volume) share a formal characterisation but were specified by only a small number of people.

The Standard Upper Ontology (SUO)[6] of the IEEE is perhaps the only example of a community-driven ontology building building effort aimed at a formal characterisation of a broad range of concepts, even though its strong points lie in predominantly physical domains as engineering and physics. Unfortunately, its development stalled when the working group was unable to resolve several hot issues via the mailing list. At the heart of this debate lay the proposal for a Standard Upper Merged Ontology[7] (SUMO) by Niles and Pease (2001), who had gathered several existing proposals (e.g. the top ontology of Sowa) into a single framework. Unfortunately, SUMO was rejected as a consensus ontology, and recast as Suggested Upper Merged Ontology, again developed by only a handful of people.

Surprisingly enough, the same holds for some of the most popular standard vocabularies used across the web today. FOAF, Dublin Core and RSS were initially specified by a small group of people. Although they tend to be limited in scope and have relatively lightweight semantics, their development as standards only happened after they were already adopted by scores of users. Because of this, the specification of e.g. revisions to Dublin Core *were* a community effort: several parties had built-up a vested interest in the language. The result is a collaborative process which is just as slow going and burdensome as was the development of the infinitely more rigourous and extensive SUO.

[6] See `http://suo.ieee.org/`

[7] See `http://www.ontologyportal.org/`

On Weak Hearted Surgeons

To recapitulate, the emphasis on standardisation did not have the expected result. True, several ontologies are currently used as standard vocabularies, but these were not developed through a *process* of standardisation, rather they were *adopted* as standards or reflect an already existing standard. It is furthermore unclear whether ontology engineering as social activity has any beneficial effect on the quality of the resulting ontology. Are ontologies based on consensus 'better' ontologies? Experience shows that this is not necessarily the case, if only because different parties may not even agree on the purpose of the ontology. For instance, if the ontology is to "facilitate knowledge exchange between applications" this may be interpreted as "the ontology should cover the knowledge expressed by my application" by individual contributors. The result is that politics get in the way of quality, and definitions are tweaked for the sake of consensus. This is evident in e.g. FOAF which includes a wide range of properties to refer to the name of persons: foaf:name, foaf:givenname, foaf:firstName, foaf:family_name and foaf:surname. But it was also what eventually stopped SUO progress; a consensus ontology was drafted which was deemed unacceptable by some on theoretical grounds.

A second impediment to ontology construction as social activity, is that contrary to standardisation in general, we are not dealing with a level playing field. The formal definition of terms is often quite technical, which means that not every interested party can fully participate in the definition process. Who would like to confront a subject matter experts with the intricacies of the interaction between global domain and range restrictions and existential restrictions on classes? Furthermore, the activity is otherwise constrained as well: a formal representation language typically puts limitations on what *can* be expressed, the most general restriction being consistency. For a long time it was held that ontology engineering required ontologies to be specified at the knowledge level, rather than directly in a representation formalism. Of course, the knowledge management view of ontology is compatible with this perspective, and most ontology engineering methodologies do not really pay much attention to this language bias. However, they do take into account the effects of order dependence in taxonomy construction.

5.3 Categories and Structure

Building an ontology starts from a general idea of its purpose and scope. The way in which this overview is translated into a concrete set of concept definitions is not trivial. In particular, the process is order dependent and traditional top-down and bottom-up approaches introduce a bias. Working top-down by starting with a set of general terms, as traditionally practised in metaphysics, risks imprecision or rework: more specific concepts may not directly fit the general ones and provide an almost unlimited source for revisions to the top level. On the other hand, naïve bottom-up specification risks the definition of many detailed concepts that turn out to be unimportant, or incompatible at a later stage.

Hayes (1985) proposes an approach to the development of a large-scale knowledge base of naive physics. He rejects top-down construction in favour

Superordinate	Animal	Furniture
Basic level	Dog	Chair
Subordinate	Retriever	Rocker

Table 5.1: Examples of basic-level concepts (Lakoff, 1987, p.46).

of the identification of relatively independent *clusters* of closely related concepts. This allows one to break free of the often demanding rigour of taxonomic relations that inevitably forms the backbone of any ontology. By constraining (initial) development to clusters, the various – often competing – requirements for the ontology are easier to manage. Clusters can be integrated at a later stage, or used in varying combinations allowing for greater flexibility than monolithic ontologies. The idea of clustering information was later adopted and refined by Uschold and King (1995), who propose a *middle-out* approach where construction proceeds up and downwards from a relatively small set of *basic* notions. These basic notions are grouped into clusters, and should be defined before moving on to more abstract and more specific terms.

5.3.1 A Basic Level?

The notion of this *basic level* follows the theory of categorisation in human cognition of Lakoff (1987). It is based on the idea that certain concepts are not merely understood intellectually; rather, they are *pre-conceptual*, used automatically, unconsciously, and without noticeable effort as part of normal functioning. This *functional embodiment* means that concepts used in this way have a different, and more important psychological status than those that are only thought about consciously. Basic level concepts are functionally and epistemologically primary with respect to (amongst others) knowledge organisation, ease of cognitive processing and ease of linguistic expression. This *basic-level primacy* makes that we generally use categories at this level to compose more complex and specific categories, and can create more general categories by explaining their differences and correspondences. In other words, categories are organised so that the categories that are cognitively basic are 'in the middle' of a taxonomy. Generalisation proceeds 'upwards' from this basic level and specialisation proceeds 'downwards' (see Table 5.1). In fact, the basic level transcends multiple levels of abstraction, e.g. 'bird' is basic where 'mammal' isn't. Research in linguistics, most notably Ogden (1930)'s *Basic English* – the inspiration for Orwell's fictional language *Newspeak* in his novel '1984' – has indicated that this holds for words as well: about 90 percent of the English vocabulary can be meaningfully described using the other 10 percent; some 840 words.

This seems quite a sensible approach: it is easy to see how basic level concepts can be used as cognitive primitives for expressing more general and specific concepts. A basic level concept indicates the most prominent characteristics of a specific concept as in 'A retriever is a dog that is bred to retrieve game for a hunter', and can be used as prototypical examples in the definition of more general concepts: 'Animals are dogs, cats, mice, birds, ...'.

Lakoff's basic categories give us intuitive building blocks to derive and cluster more complex notions. They carry a lot of weight and form the conceptual foundation of an ontology. However, the primitive, pre-conceptual

endless loop, n. See loop, endless.
loop, endless, n. See endless loop.

Table 5.2: The difficulty to define basic concepts, from Pinker (2007, p.12).

nature of the basic level means that representing basic categories *themselves* is certainly not as straightforward. Our knowledge of basic concepts such as dog, chair, hammer, father is largely implicit and hard to break down into smaller components, exactly because they are primitive.

Consider the following definitions (taken largely from Wikipedia):

- A *dog* is an animal that has four legs and barks. Animals are dogs, cats, mice, birds.... A leg is something you use to walk with. Barking is the sound dogs make.
- A *chair* is a kind of furniture for sitting, consisting of a back, and sometimes arm rests, commonly for use by one person. Chairs also often have four *legs* to support the *seat* raised above the floor.[8] Seat can refer to chair.[9] A back is something you can lean against. An arm rest is something you can rest your arm on while sitting.
- *Father* is defined as a male parent of an offspring.[10] A parent is a father or mother; one who sires or gives birth to and/or nurtures and raises an offspring. [11]
- A *hammer* is a tool meant to deliver blows to an object. The most common uses are for driving nails, fitting parts, and breaking up objects.[12] [...] a nail is a pin-shaped, sharp object of hard metal, typically steel, used as a fastener. [...] Nails are typically driven into the workpiece by a hammer.[13]

It seems that the more basic a concept is, the harder it is to cast its description in terms of other concepts (cf. the quote from Pinker (2007) in Table 5.2). Often definitions of basic concepts will be self-referential and form a tautological trap from which it is very hard to escape. We have seen in Chapter 3 that the definition of a concept in knowledge representation languages such as OWL can only be expressed as (restrictions on) relations with other concepts. This means that, inevitably, the meaning of a concept is determined by context. The basic level theorem does not take this into account. Although basic concepts are *perceived* as primitive, they too can only be defined in this manner: that is, either in terms of other basic concepts or (inevitably) by reference to more generic or specific concepts.

The main problem of traditional top-down and bottom-up approaches was that the commitment to a set of generic or specific concepts introduces a bias that may lead to an ill fit between the resulting ontology and the intended

[8] From `http://en.wikipedia.org/wiki/Chair`
[9] http://en.wikipedia.org/wiki/Seat
[10] From `http://en.wikipedia.org/wiki/Father`
[11] From `http://en.wikipedia.org/wiki/Mother`
[12] From `http://en.wikipedia.org/wiki/Hammer`
[13] From `http://en.wikipedia.org/wiki/Nail`

representation. But it turns out that this drawback is only partially lifted by the middle-out approach. Basic level concepts introduce a similar bias as they too can only be defined in context with other concepts. This context is largely implicit and inaccessible because of the cognitive primitive nature of the basic level.

In short, basic concepts cannot be defined prior to other concepts as the middle-out approach suggests. It is rather the other way around. Basic level concepts give a 'feel' of relatedness and context. They are an ideal entry point to carve up the domain into relatively independent clusters, and play an important role in the definition of less basic notions. Lakoff's basic concepts should therefore be seen as being at the core of Schank and Abelson's scripts and Minsky's frames (Section 2.2.4). Regarded as independent entities they are atomic. They do not carry any meaning, but rather function as meaning providers to surrounding notions. Their own meaning, however, can only really emerge through their use in other definitions.

5.3.2 Beyond Categories

To be frank, this is as far as ontology engineering methodologies go. They do not really venture beyond well-meant, but gratuitous advice such as 'be sure of what you want before you start', 'document your decisions well, you might forget them later' and 'if the problem is too big, chop it up into smaller chunks'. The decomposition into various phases therefore plays the role of the user manual no-one ever reads.

Admittedly, several suggestions *are* genuinely helpful and convey a deeper understanding of what knowledge engineering entails. Ontology engineering is often called an art or craft, rather than a science or proper engineering field such as mechanical engineering or software engineering (Fernández-López and Gómez-Pérez, 2002). Although this claim is often cited, it is never really substantiated. What it does hint at, however, is that somewhere along the line something magical happens. Apparently some inaccessible mental skill only found in an experienced knowledge engineer can turn some bit of knowledge into an adequate representation. The adoption of Lakoff's basic level by Uschold and King was a conscious move to apply insights from cognitive science to lay bare a part of this process.

We have seen that basic level concepts can help in the formation of clusters and the construction of definitions of non basic concepts without the straitjacket of top-down and anarchy of bottom-up development. However, there is no methodological support for the way in which the definitions in an ontology are specified in practice. Ontology construction is to proceed 'upward' and 'downward' from the basic level, but how?

5.4 Ontology Integration

Ontologies are rarely constructed without at least a pre-existing conception of that which is to be represented. They can be meant to directly express an existing theory, as we have seen in the biomedical domain, or capture some body of expert knowledge. Even in the least restricted case, where an ontology is meant to represent common knowledge, that which is to be represented is

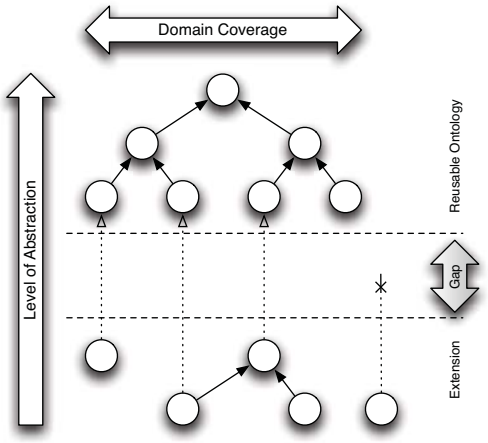

Figure 5.2: Dimensions in ontology reuse

relatively well understood. Nonetheless, ontology development is hard, for how to reliably translate our knowledge into a set of formal axioms?

As discussed, one of the main roles of ontologies is to ease knowledge acquisition by providing readily usable concept definitions. The same bootstrap can be applied to ontology development itself; existing ontologies can be a viable resource for ontology construction in the form of predefined concepts and relations. These definitions convey ontological commitments that capture important ontological choices (Masolo et al., 2003). When these choices are in concert with our requirements, adopting and extending an existing ontology relieves us from the burden of micromanaging these choices ourselves.

One solution is thus to rely on an already existing ontology to provide initial structure: the use of a more general ontology to support knowledge acquisition. This is in fact partly addressed in the ontology integration step of Uschold (1996); Uschold and Grüninger (1996). Reusing and extending an existing ontology solves the structure problem of blank slate ontology development as the existing ontology can be used as a *coat rack* for new concept definitions (Schreiber et al., 1995).

Klein (2001) gives an overview of several issues related to ontology reuse and distinguishes between ontology *integration*, where two or more ontologies are combined into a single new ontology, and *reuse* where the ontologies are kept separate. Both approaches require the ontologies to be *aligned*; the concepts and relations that are semantically close should be related via equivalence and subsumption relations. Since Klein focuses on methods to combine multiple already existing ontologies, this alignment may involve significant modification to the ontologies involved in case they overlap. However, it is not uncommon to modify a source ontology in its reuse as basis for an entirely new ontology. For instance, Pinto and Martins (2000) stress a thorough evaluation of candidates for reuse, and see the modification of those ontologies on the basis of this evaluation as an integral part of ontology development.

Modifications can be necessary if the combination of ontologies results in a heterogeneous system where mismatches may exist at either the *language level*

or the ontological or *semantic* level (Klein, 2001; Visser et al., 1997). Where the former concern incompatibility with respect to how concepts and relations *can* be defined, the latter signals differences in the ontological assumptions about the represented domain. Language level mismatches are not merely syntactic, such as in the incompatibility of the KRSS lisp-style syntax with the RDF/XML syntax of OWL, but may be more fundamentally related to the expressiveness of different languages (Chalupsky, 2000). Since a knowledge representation formalism can be uniquely identified by its position in the traditional trade-off between expressiveness and tractability identified by Levesque and Brachman (1987),[14] the alignment of two representations in different languages invariably involves either choosing one position over the other, or finding a common middle-ground. In the worst case, when translating to a less expressive language this will lead to loss of information. Translating to a more expressive language is less problematic, but will nevertheless require mapping and rewriting of many knowledge representation idioms. However, in many cases the languages overlap; e.g. one language may be more expressive in one area and sport qualified cardinality restrictions, where the other does not do so, but allows for property chain inclusion axioms.

Semantic mismatches are more subtle, and are sometimes even the main reason for starting the alignment. For instance when two ontologies differ in coverage and granularity, their conjunction will increase both.[15] Ontologies may also differ in their ontological stance (Masolo et al., 2003), e.g. choosing a 4D over a 3D perspective on identity persistence through time (Haslanger, 2003), or with respect to more practical modelling paradigm and conventions (Chalupsky, 2000), such as interval versus point-based representation of time, or representing property values as constants or datatype values. Resolving mismatches between ontologies can to some extent be facilitated using techniques for (semi) automatic ontology merging and alignment, such as in the PROMPT system of Noy and Musen (2000).[16]

Although the problem of integrating existing ontologies is more intricate than direct ontology reuse for the development of a new ontology, the possible mismatches give insight in the kinds of things one should look for when choosing a suitable candidate for reuse. This choice thus depends, besides on its overall quality, on four suitability factors: *domain coverage* of the ontology, its *level of abstraction* and *granularity*, its *representation language* and the *level of formality*. Figure 5.2 shows the interaction between these factors.

Domain coverage corresponds to the breadth of the reused ontology. Definitions in an ontology are reused by linking into its structure: axioms in the new ontology are expressed in terms of the existing vocabulary. If the domain coverage of the reused ontology is sufficient, all new definitions can be connected in this a way. Limitations in coverage require either one of two measures: the addition of a new concept at a higher level of abstraction, or the reuse of another ontology that *does* provide the necessary definition. Both solutions are not without problems. First of all, extending the new ontology with more general concepts violates the principle of *stratification*, the idea that different levels of abstraction should be separate, and reuse should accumulate in lay-

[14] See also Section 3.4

[15] This holds especially when constructing an entirely new ontology, as it does not (yet) *have* coverage or granularity.

[16] See Klein (2001) for a comprehensive overview.

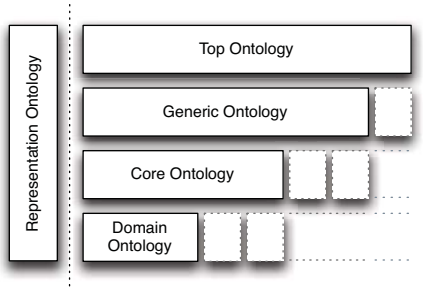

Figure 5.3: Ontology types based on van Heijst et al. (1997)

ers. Adding generic concepts to the new ontology may lead to a necessity to augment already existing concept definitions in the reused ontology. For instance, adding the common abstract notions of 'occurrent' and 'continuant' when reusing an ontology that does not make this distinction requires disjointness axioms between existing concepts. This may be quite sensible from the standpoint of direct reuse, but breaks compatibility with other ontologies that reuse the same ontology. Secondly, extending coverage by reusing another ontology is only feasible in case both reused ontologies are aligned with each other.

A typical way in which the coverage of ontologies is maximised is by increasing its level of abstraction. It would thus seem that very general ontologies are ideal candidates for reuse. Nonetheless, a higher level of abstraction increases the gap between concept definitions in the reused ontology and the level at which new concepts are ideally defined. Reusing an abstract ontology means that the concepts it defines provide little direct structure for new concept definitions. It thus requires extra work to bridge this gap.

Lastly, if the ontology is specified in a suitable representation language, the inheritance of properties and other implicit knowledge can be used to check not only consistency but also the extra-logical quality of the ontology extension: whether what is derived about new classes and properties makes sense (Section 5.2). Note that a more formal representation language has only a partial effect on level of formality. An ontology in OWL 2 DL that specifies only named classes and subclass axioms is in fact only in the $\mathcal{AL}$ fragment of DL, whereas an ontology in the same language that uses some of its more expressive features may be in full $\mathcal{SROIQ}(D)$. Corresponding representation languages signal compatibility of inference – inferences on definitions in the reused ontology propagate to new definitions – whereas a higher level of formality increases the *number* of inferences, and thus usability, but also constrains coverage to only those domains compatible with its entailments.

5.4.1 Types of Ontologies: Limits to Reuse

Ontology reuse received quite a lot of attention, most notably in the construction of ontology libraries (Breuker and Van De Velde, 1994; van Heijst et al., 1997). Rather than by purpose – as in Chapter 4 – van Heijst et al. distinguishes ontology types by the level of abstraction of their content (See Figure 5.3). This

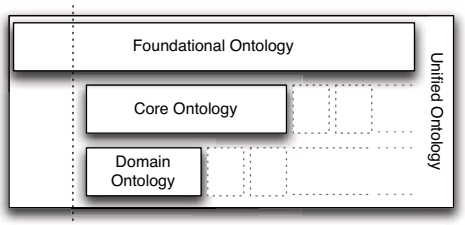

Figure 5.4: Contemporary ontology types

categorisation was based on the idea that the possibility for ontology reuse is influenced by the level of domain-dependence of the ontology.

van Heijst et al. (1997); Valente and Breuker (1996) distinguish five types of ontologies that provide a layered approach to reuse, and signal compatibility between knowledge based systems. A *top ontology* specifies highly generic categories and relations that are relevant to all domains. *Generic ontologies* specify general notions that are applicable across multiple domains, such as time, space, causality etc. A *core ontology* is a general ontology of application domains such as law, medicine and biology, and specifies a minimal set of concepts needed to describe entities in the domain. *Domain ontologies* specify the conceptualisation of a particular domain. Examples are the Linnaean taxonomy of species or an ontology of vehicles. An *application ontology* is an ontology that captures the ontological commitments of a single application. In this list, the representation ontology, of which the Frame Ontology is an example, is the odd-one out as it operates at a meta level with respect to concepts in an ontology (Section 4.4).

The distinction between these types is given by a theoretical consideration to organise the ontologies in an ontology library in a modular fashion and at strictly separate levels of abstraction. For instance, the core ontology type was initially proposed by van Heijst et al. (1997) as a means to better index domain ontologies stored in the library. Similarly, the specification of an application ontology is a fixed step in the *Common*KADS methodology for knowledge based systems development (Schreiber et al., 2000): a domain ontology is refined into an application specific ontology, which is then implemented in a knowledge based system. Although a similar modular organisation can be found in ONTOLINGUA, and more recently in the strict editorial process in OBO Foundry, the distinction is artificial. In practice ontologies rarely fit neatly into one of the categories. Contemporary ontologies can be roughly distinguished as *unified*, *foundational*, *core* or *domain* ontologies (Figure 5.4). Each of these ontology types holds a different position with respect to the trade off between the factors of coverage, abstractness and level of formality.

Unified Ontologies

Unified ontologies are ontologies that aim to present a 'grand unifying theory'; they cover both highly abstract as well as concrete concepts from a wide range of domains. Perhaps the most typical unified ontology is the SENSUS ontology of Swartout et al. (1996), which was developed explicitly with the purpose of reuse in mind. It is a large (50k) skeletal ontology with a broad coverage,

and contains both abstract and relatively concrete concepts from a variety of sources. Swartout et al. (1996) describe a general methodology where ontology construction should proceed by selecting and refining relevant parts of the SENSUS ontology. This way, it was thought, one could avoid "stovepipe" ontologies that can be extended vertically, i.e. along the specific-generic axis, but not horizontally, across domains.

However, formal definitions would impose interpretations on concepts that constrain their usability. For this reason, the ontology was relatively lightweight to maximise domain coverage, it is a knowledge management ontology. The Wordnet system of Miller (1990), which was in fact part of SENSUS, is often used in a similar way to populate ontologies. In the end, SENSUS did have some utility as source of inspiration for more specific ontologies, but could not enforce particular inference nor guarantee consistency between its different extensions. Another example of a unified ontology is the CYC ontology of Lenat et al. (1990); Lenat (1995), which is discussed in more detail in Section 6.2.1. In contrast to the SENSUS ontology, CYC is specified using an expressive formal language.

Concluding, unified ontologies are not the most suitable for direct reuse. When lightweight they provide little guidance in defining new concepts, and when rigourously formal their broad coverage introduces significant computational overhead.

Foundational Ontologies

Foundational ontologies or *upper* ontologies, are hybrid ontologies that combine top ontologies with the description level of a representation ontology. They have a philosophical disposition and focus on the basic categories of existence. Examples are SUO/SUMO (Niles and Pease, 2001), DOLCE (Masolo et al., 2003), BFO (Grenon, 2003) and the General Formal Ontology (Herre et al., 2006).[17] The foundational ontology ascribes to a role similar to that of unified ontologies in that it is aimed at coverage of a broad range of domains. Foundational ontologies find their roots in the philosophical discipline of formal Ontology.[18] In fact, most philosophical ontologies are foundational.

Contrary to unified ontologies, foundational ontologies are based on the stance that 'abstract is more': their applicability to a multitude of domains is ensured by defining general, abstract notions rather than by the direct inclusion of a large number of relatively specific concepts. Although the foundational approach ensures rigorous ontological definitions and broad coverage, this comes at a cost. First of all, it is to be expected that the rather abstract nature of foundational ontologies makes them less suitable candidates for knowledge acquisition support in the development of concrete domain ontologies. For instance, the developer of a domain ontology of pet animals has little to gain from such philosophically inspired classes as a:FiatObjectPart and a:GenericallyDependentContinuant in BFO. The distance between concepts in the foundational ontology and concrete domain concepts is just too big (cf. Figure 5.2).

[17] Note that although DOLCE was not initially developed as foundational ontology, it is often used as such. See Chapter 6.

[18] SUMO is an exception to the rule, as it only partly relies on the top ontology of Sowa (2000) and provides relatively concrete definitions in the physical domain.

A related issue is that reasoning over the top level categories in foundational ontologies may not necessarily have a direct bearing on practical reasoning problems because of a possible proliferation of inferences over abstract categories. Furthermore, their philosophical disposition makes that they are traditionally specified in a form of first order logic, and define meta-level categories at the ontological level of Guarino (1994). Although OWL compatible versions are sometimes available, the limited expressiveness of this language means that such versions are ontologically less strict, and are not entirely faithful to the full ontology.

Core and Domain Ontologies

Core ontologies balance coverage and abstractness by restricting the scope to a particular area of expertise. Examples of such ontologies are the LRI-Core and LKIF Core ontologies of Breuker and Hoekstra (2004a); Breuker (2004) and Breuker et al. (2007); Hoekstra et al. (2008, this volume) respectively, the Core Legal Ontology and Fishery Ontology of respectively Gangemi et al. (2005) and Gangemi et al. (2004), and more domain independent ontologies such as the COBRA ontology of business processes analysis (Pedrinaci et al., 2008), or for information integration (Doerr et al., 2003). Although the scope restriction limits the possible situations in which core ontologies can be used, it has several advantages. A core ontology is more likely to provide definitions of concrete concepts which are more intuitive candidates for reuse, and can sanction more relevant inference.

On the other hand, the concreteness of core ontologies is not a necessary given. Because important domains such as Law and Medicine are highly specialised and theoretical, they include very abstract concepts one does not readily find in generic ontologies. Examples are the legal ontologies of Mommers (2002); Lehmann (2003); Rubino et al. (2006) that reflect stances in legal theory and cover such abstract concepts as rights, duties, liability and legal causation (see also Chapter 6).

Domain ontologies extend core (or foundational) ontologies with more concrete categories that are of direct relevance to inference in a specific domain. To give an example, the Fishery Ontology of Gangemi et al. (2004) is an extension of the DOLCE ontology and defines such concepts as Fishing_Area, Aquatic_Organism and Catching_Method that can be refined to Aberdeen_Bank, Herring and Pelagic_Trawling in a domain ontology on North Sea fishing. Similarly, a domain ontology of criminal law would introduce subclasses of Crime specific to a particular jurisdiction and iterate specific felonies and misdemeanours such as Manslaughter and Petty_Theft.

5.4.2 Safe Reuse

It can be necessary to reduce the expressiveness of a reused ontology for it to fit a different, more restricted knowledge representation formalism. This practice is often seen in cases where foundational ontologies are applied in a Semantic Web context. For instance, in the SWIntO ontology of Oberle et al. (2007)[19], which is a combination of SUMO and DOLCE, the expressiveness of SUMO

[19] SWIntO: SmartWeb Integrated Ontology

is constrained by stripping it of most (if not all) of its axioms, leaving only a bare-bone subsumption hierarchy. However, this weakens the role of an ontology as source for ontological choices and design patterns as any modification of an ontology compromises the commitment of the ontology to its original conceptualisation.

To illustrate this point, consider an ontology as a set of ontological commitments. These commitments are comprised of a set of (non-logical) symbols representing the entities the ontology claims to exist, the *signature* of the ontology, and a set of *ontological statements* about these entities.[20] A difference in either of these sets of commitments makes a different claim about reality – or what can exist in a knowledge base – and thus denotes a different ontology.

Removing a symbol or statement means that what is reused is no longer the original ontology, but rather a scaled-down version. Because in that case, no direct match of the ontological commitments of the source ontology and the new ontology can be established, this practice compromises the purpose of ontology reuse. Namely, to ensure compatibility between the different ontologies that reuse and commit to the same, more general ontology (Gruber, 1994). This traditional reuse is only possible by full *formal* reuse as opposed to *informal reuse*:

Definition 5.4.1 (Informal Reuse) *The* informal reuse *of one ontology $\mathcal{O}'$ by another ontology $\mathcal{O}$ requires (at the least) some syntactic correspondence between the presence of named entities in the signatures of both ontologies: $\mathsf{Sig}(\mathcal{O}) \cap \mathsf{Sig}(\mathcal{O}') \neq \emptyset$.*

In other words, informal reuse does not imply a *formal* commitment to the theory expressed by the reused ontology: the commitment exists only with respect to part of the signature of the reused ontology, and cannot be established through formal means. Of course, informal reuse can range from a very loose correspondence to a tight integration. Informal reuse can still be quite rigorous and close to the original interpretation by a commitment to other than formal requirements, such as a specification in natural language.

For a more formal notion of reuse, we further restrict the definition of reuse by adopting an imports paradigm: the reusing ontology should 'import' all ontological commitments of the reused ontology. Ontology import is generally defined using two closure axioms (Motik et al., 2009):[21]

Definition 5.4.2 (Imports Closure) *The transitive* imports closure *of an ontology $\mathcal{O}$ consists of $\mathcal{O}$ and any ontology that $\mathcal{O}$ imports.*

Definition 5.4.3 (Axiom Closure) *The* axiom closure $\mathcal{O}_\cup$ *of some ontology $\mathcal{O}$ is defined as the smallest set that contains all axioms from each ontology $\mathcal{O}'$ in the* imports closure *of $\mathcal{O}$.*

These closures are syntactic constructs that gather together the set of all axioms, formal ontological statements, that express an ontology's commitments.

[20] Formally, the signature of an ontology is the set of all non-logical symbols, i.e. all entities in the ontology excluding the symbols of the knowledge representation language and sentences over the non-logical symbols. Note that ontologies specified in a non-formal language can be said to have a similar signature, albeit not as explicit.

[21] Though these definitions lie at the heart of OWL 2's import mechanism (Section 3.5.2), they are not necessarily particular to that language.

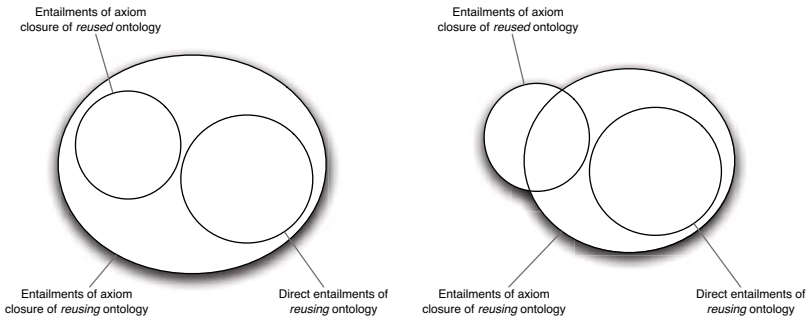

Figure 5.5: Relation between entailments in full and partial formal reuse

The partial *semi formal* reuse of e.g. SUMO by SWIntO, i.e. the commitment to just the taxonomic hierarchy of SUMO without considering its complex class axioms and properties, can be expressed in terms of these closures in a straightforward manner:

Definition 5.4.4 (Semi Formal Reuse) *An ontology $\mathcal{O}'$ is reused by ontology $\mathcal{O}$ through* partial semi formal reuse *if the intersection of the axiom closures $\mathcal{O}'_\cup$ and $\mathcal{O}_\cup$ is non empty:*

$$\mathcal{O}'_\cup \cap \mathcal{O}_\cup \neq \emptyset$$

Full semi formal reuse *is the case where the axiom closure $\mathcal{O}'_\cup$ is a subset of the axiom closure of $\mathcal{O}_\cup$:*

$$\mathcal{O}'_\cup \subseteq \mathcal{O}_\cup$$

Although this reuse ensures the inclusion in $\mathcal{O}$ of at least some axioms of the reused ontology $\mathcal{O}'$, it is only semi formal. Axioms are regarded as distinct *syntactic* entities, without taking their semantics into account. However, the axioms of different ontologies may interact, and cannot be considered separately. Even in the case of full semi formal reuse, combining previously unrelated axioms can have surprising effects.

When some ontology $\mathcal{Q}$ is imported by an ontology $\mathcal{P}$, the semantics of the resulting axiom closure $\mathcal{P}_\cup$ is determined by the union of the axioms $\mathcal{P}_\cup = \mathcal{P} \cup \mathcal{Q}_\cup$. Although $\mathcal{P}$ imports all axioms from $\mathcal{Q}_\cup$ this is no guarantee that the ontological commitments of $\mathcal{P}_\cup$ are compatible with that of $\mathcal{Q}_\cup$. The intention of reuse is that axioms in $\mathcal{P}$ interact with the axioms in $\mathcal{Q}_\cup$ in such a way that in $\mathcal{P}_\cup$ new entailments may hold for the axioms in $\mathcal{P}$. However, the meaning of terms in $\mathcal{Q}_\cup$ may change as a consequence of the import as well (Cuenca Grau et al., 2007).

For instance, consider the following axioms in $\mathcal{Q}$:

$$\begin{aligned} \mathsf{Bird} &\sqsubseteq \mathsf{has\_part}\ \textbf{exactly}\ 2\ \mathsf{Wing} \\ \mathsf{Penguin} &\sqsubseteq \mathsf{Bird} \end{aligned}$$

Because of the first axiom, we can infer $\mathsf{Penguin} \sqsubseteq \mathsf{has\_part}\ \textbf{exactly}\ 2\ \mathsf{Wing}$: because penguins are birds, they also have exactly two wings. Suppose that $\mathcal{P}$

imports $\mathcal{Q}$ and refines the definition of Bird with the axiom:

$$\mathsf{Bird} \sqsubseteq \mathsf{has\_ability}\ \textbf{some}\ \mathsf{Flying}$$

Thus, where $\mathcal{P}$ entails that birds can fly, $\mathcal{P}_\cup$ entails that birds can fly and have two wings. However, this seemingly innocent (and useful) extension will also allow us to infer that penguins can fly: $\mathsf{Penguin} \sqsubseteq \mathsf{has\_ability}\ \textbf{some}\ \mathsf{Flying}$. Without altering $\mathcal{Q}$ directly, adding a single axiom in $\mathcal{P}$ has changed the meaning of terms in the ontology: its ontological commitments in the context of $\mathcal{P}$ differ from its explicit commitments.

Of course, it is easy to come up with more far reaching modifications to $\mathcal{Q}$, such as adding disjointness axioms or meta modelling. This is problematic from a traditional reuse perspective, as some other ontology $\mathcal{P}'$ may reuse and extend $\mathcal{Q}$ as well, and introduce yet other interactions with the axioms of $\mathcal{Q}$:

Definition 5.4.5 (Ontology Interaction Problem) *The* ontology interaction problem *is the general problem that if an ontology reuses another, its ontological commitments may affect the commitments of the reused ontology and thereby undermine its compatibility with other ontologies that reuse the same ontology.*

How then to make sure that $\mathcal{P}$ and $\mathcal{P}'$ are compatible, and information exchange between two systems respectively committing to these ontologies will be possible? It is clear that because of the syntactic nature of the axiom closure, the union of two closures does not give us an orderly definition of reuse. At the least, formal reuse of an ontology $\mathcal{O}'$ by an ontology $\mathcal{O}$ should not affect the original entailments of $\mathcal{O}'$, and if any part $\mathcal{O}'' \subseteq \mathcal{O}'$ is (for some reason) left outside the axiom closure of $\mathcal{O}$, its entailments should neither be affected by $\mathcal{O}$ nor vice versa. In other words, any partial reuse of $\mathcal{O}'$ may only ignore that part $\mathcal{O}''$ that is wholly disconnected and irrelevant to the extension $\mathcal{O}$.

Ghilardi et al. (2006) introduce a notion of *safe reuse*, direct formal reuse of a signature in an ontology, based on the definition of *conservative extension* in description logics:

Definition 5.4.6 (Conservative Extension) *An ontology $\mathcal{O}$ is an* S*-conservative extension of $\mathcal{O}_1$, if $\mathcal{O}_1 \subseteq \mathcal{O}$ and for every axiom α expressed using signature* S*, we have $\mathcal{O} \models \alpha$ iff $\mathcal{O}_1 \models \alpha$. The ontology $\mathcal{O}$ is a* conservative extension *of $\mathcal{O}_1$ if $\mathcal{O}$ is an* S*-conservative extension of $\mathcal{O}_1$ with respect to the signature of $\mathcal{O}_1$.*

In other words, the axiom closure $\mathcal{P}_\cup$ is only a conservative extension of $\mathcal{Q}_\cup$ if no axioms are implied by $\mathcal{P}_\cup$ over symbols in the signature of $\mathcal{Q}_\cup$ that are not also implied by $\mathcal{Q}_\cup$ itself. This notion can be straightforwardly applied to establish that in our case, $\mathcal{P}_\cup$ is not conservative with respect to $\mathcal{Q}_\cup$ because in $\mathcal{P}_\cup$ penguins can fly, and in $\mathcal{Q}_\cup$ they cannot.

An ontology $\mathcal{O}$ is *safe for a signature* if no other ontologies $\mathcal{O}'$ using that signature can exist for which $\mathcal{O} \cup \mathcal{O}'$ is not a conservative extension of $\mathcal{O}'$ (Cuenca Grau et al., 2007). Unfortunately, Cuenca Grau et al. also show that determining whether some set of axioms (an ontology) is safe with respect to a signature is undecidable for expressive description logics.

It is clear that formal reuse of an ontology must be complete with respect to the logical consequences of axioms expressed in that ontology (Figure 5.5). But not only must the reusing ontology be consistent with the source ontology, it also should be a conservative extension. Note, however, that formal

ontology reuse does not require the existence of an imports relation between two ontologies, but rather ensures compatibility between their commitments. Nor does formal reuse necessarily require a compatibility between the representation formalisms used by both ontologies, rather determining whether an extension is conservative easily becomes undecidable for arbitrarily differing knowledge representation languages.

Recently, several tools were developed to support *partial formal reuse* by means of ontology module extraction Konev et al. (2008); Jimenez-Ruiz et al. (2008). In this case, some ontology may $\mathcal{O}$ reuse part of the axioms in an ontology $\mathcal{O}'$ provided that it is safe with respect to the signature of the reused axioms. Module extraction is a process by which exactly those axioms that affect the semantics of axioms in the reusing ontology are extracted to form a coherent whole. Of course, this is only of practical use when the reused ontology has limited density, i.e. when axioms in the ontology are relatively independent. Furthermore, the safety of $\mathcal{O}$ with respect to the signature of axioms from $\mathcal{O}'$ for expressive DLs can only be determined using *locality conditions* (Cuenca Grau et al., 2007; Jimenez-Ruiz et al., 2008), or for languages that have favourable properties (Konev et al., 2008).

5.4.3 Possibilities for Reuse

The previous section discusses the general problem of reusing knowledge expressed by some ontology. The original ideals of direct ontology reuse between systems expressed by Gruber (1994) and others, seem to be only attainable given the following restrictions:

Restriction of Non-Modification
Reused ontologies should not themselves be modified in any way.

Restriction of Formal Reuse
Ontological correspondence between systems can only be ensured if all ontologies are expressed in a formal knowledge representation language.

Restriction of Compatibility
Formal languages used to express the ontologies that share ontological commitments for the purpose of knowledge reuse, should be compatible in such a way that the entailments of one ontology can be checked for consistency with any of the other ontologies.

Restriction of Complete Reuse
A reusing ontology should directly import the ontological commitments of the (part of the) reused ontology it commits to, i.e. it should import both its signature and ontological statements.

Restriction of Safe Reuse
A reusing ontology should at least be a conservative extension of the reused ontology, and it should ideally be safe with respect to the signature of the axioms it reuses.

Given these restrictions, we have an extra, formal tool for assessing the suitability of ontologies for reuse that augments the more conceptual considerations discussed in Section 5.4.1, and the distinction between knowledge management, representation and formal ontologies of Chapter 4. Table 5.3 shows

Type	Purpose	Level	Meta	Expressiveness
Unified	KM/KR	abstract/concrete	yes/mixed	FOL, informal, restricted
Foundational	FO	abstract	yes	FOL
Core	KR/FO	abstract/concrete	no	FOL, restricted
Domain	KM/KR	concrete	no	restricted

Table 5.3: Characteristics of ontology types and purposes

some of the typical characteristics of the types of ontologies introduced in that section. Unified ontologies operate at both abstract and concrete levels of description, involve meta modelling and are expressed using languages of varying expressiveness. Foundational ontologies are limited to the definition of abstract notions using (variants of) first order logic (FOL) at a meta level. Some have versions expressed using a more restricted language. Core ontologies do not use meta modelling in the definition of concepts, and mix abstract and concrete levels of description. They are usually, but not always represented using the restricted expressiveness of knowledge representation languages. Domain ontologies offer the most concrete concept definitions in languages of restricted expressiveness.

The use of different formal languages reflects the difference between an abstract more philosophically inspired perspective and the concrete practical perspective of knowledge engineering. In Section 4.4 this distinction was already discussed in more detail. The necessity of formal reuse shows a discontinuity in the layered approach for reuse aspired to by Valente and Breuker (1996); van Heijst et al. (1997); Schreiber et al. (2000): abstract and concrete ontologies do not *connect*. Foundational ontologies can only be reused by ontologies at a concrete level if they use the same expressive formalism. However, expressive languages such as KIF and CommonLogic are generally undecidable and no formal nor pragmatic means to ensure safe reuse are available.[22] In part this is overcome by the availability of variants of foundational ontologies in restricted, decidable languages, but these do not capture the full set of ontological commitments of the original.

5.5 Ontological Principles

Besides being a jump start in the form of a conceptual coat rack, the reason for reusing some ontology during ontology construction is the inclusion and adoption of its ontological choices. If these choices are explicit in the form of ontological commitments of the reused ontology, they can be directly applied in the construction of new concept definitions. Where the reused ontology is sufficiently dense in its definitions, it may function as a mould that prevents the ontology engineer from inadvertently constructing ontological oddities. However, as discussed the previous section, the direct reuse of pre existing ontologies is subject to several limitations. Furthermore, although formal ontological commitments may make the ontological choices explicit, it can be quite difficult to extract, and reverse engineer these choices from a bag of axioms expressed using a complex formal language. It may therefore pay to improve the accessibility of ontological choices by making explicit some of the principles

[22] CommonLogic allows for the specification of decidable dialects.

underlying the construction of ontologies. These *ontological principles* can then be applied as extra methodological guidelines during ontology construction. Where the design criteria of Gruber (1994) pose requirements for the ontology as a whole, ontological principles provide guidance in the representation of an entity as belonging to a fundamental category, i.e. what *kind* of entity it is, or as to which aspects of that entity are of ontological relevance.

5.5.1 OntoClean

Perhaps the most influential set of ontological principles can be found in the ONTOCLEAN methodology of Guarino and Welty (2002, 2004). ONTOCLEAN is a methodology for evaluating "the ontological adequacy of taxonomic relationships" and can be applied both while building a new ontology, or in its post hoc evaluation. It aims to provide a formal framework and vocabulary for justifying why a particular way of structuring a taxonomy is better than another. Central to the ONTOCLEAN methodology are the notions of *essence* and *rigidity* on the one hand, and *identity* and *unity* on the other. Once these notions are applied to two classes, they can assist to determine whether a subsumption relation between the two classes may hold.[23]

Class membership of some entity is *essential* to that entity if the entity could not exist were it not for being an instance of that class. For example, it is essential for human beings to have a mind: every human being must always be an instance of the class of things that have a mind. In fact, whether the class membership of an entity is essential depends on membership of other classes. Note that in the example, essence is conditional on class membership of the human class. For some classes this dependency is stronger than for others, and reflects its *rigidity*. All entities that are member of a rigid class depend on that class for their existence. The class membership is essential to those entities, and they can only cease to be a member of the class by ceasing to exist. A rigid class has *only* members that depend on it in this way. For instance, a naive representation may hold that having a mind is rigid, but that would mean that we would hear a distinct metaphysical '*puff*' whenever someone lost his mind, metaphorically speaking that is. Instances of the human class, on the other hand, do rely heavily on being a human for upholding their existence and it is therefore more appropriate case of rigidity: it is essential for all humans to be human beings. The rigidity of a class is the consequence of an ontological choice by the ontology engineer and thus reflects an important ontological commitment.

Non rigid classes can be inessential to some of their instances, they are *semi-rigid*, or not essential to *any* of their instances. Classes of the latter category are *anti-rigid*, typical examples of which are *roles* and *functions* such as the classes student and hammer: there is not a student that ceases to exist upon graduation. Here again, the example seems obvious, but the developer of an ontology for a library system may well want to automatically terminate library membership for graduates: a case where the rigidity of studentship is defendable.

A useful perspective to determine the rigidity of some class is the degree to which class membership is ontologically *subjective* or *objective* (Searle, 1995,

[23] The ONTOCLEAN papers use the logico-philosophical meaning of the term 'property' throughout. As this meaning differs substantially from the way in which the term is used in knowledge representation languages, the description of ONTOCLEAN will use standard KR vocabulary.

and Section 7.3). Where the existence of subjective entities depends on being felt by subjects (i.e. people), ontologically objective entities exist independently of any perceiver or mental state. This distinction helps to recognise features *intrinsic* to nature, and those *relative to the intentionality* of observers. Describing an object as a hammer specifies a feature of the object that is *observer relative*, whereas the existence of the physical object does not depend on any attitudes we may take toward it. Observer relative features are therefore ontologically subjective, and classes defined by such features are generally anti-rigid.

Deciding whether a class is essential, rigid, semi-rigid or anti-rigid in the OntoClean methodology thus puts the burden on the ontology engineer to contemplate all possible uses of the ontology. According to the methodology, the ontological commitments should be made explicit by marking elements in the ontology by means of a meta-property. This meta-property can then signal allowed use of the classes defined in the ontology. One restriction, for instance, is that rigid classes cannot be the subclass of anti-rigid classes and vice versa: the student class is therefore not a subclass of human.

Another important notion in the methodology is that of *identity*: how can we establish whether two entities are the same? Equality of entities is determined by means of *identity criteria*, and conversely if equality between entities is asserted, the identity criteria must necessarily hold. At first sight this seems very similar to the notion of essence, e.g. how in DL individuals are determined to belong to a class (See a.o. Section 7.2). However, the emphasis here lies on the properties that two individuals must have for them to be the *same individual*, e.g. for an owl:sameAs relation to hold between them. Guarino and Welty (2004) give the example of the relation between a duration and interval class. A sentence such as "All time intervals are time durations" does not entail that duration subsumes interval, as two intervals that have the same length are not necessarily the same, whereas two durations with the same length are. Although the same conclusion can be reached by giving a more conceptual account, i.e. the duration *is* the length of the interval, explicitly marking properties of classes as carrying identity criteria can be very useful to ensure innocuous reuse.

An example of an identity criterion is the inverse functional property type of OWL, which allows us to infer that two individuals are the same when they are related to the same individual via that property. For instance, two individuals that have the same social security number, must be the same person. Although a knowledge representation formalism may provide rather little in the way of built-in constructs for expressing identity criteria, these criteria can still be very useful in determining, or rather disqualifying, subsumption relations between classes.

The third notion, *unity*, refers to the property of entities as being wholes that are composed of parts (that may themselves be wholes). Unity can be used to differentiate different classes based on their mereological properties. For instance, water does not generally have parts (it carries *anti-unity*), whereas oceans do. As a rule, categories referred to using mass nouns do not carry unity, and categories that do are named by count nouns. Unity can be ensured by any of the traditional types of mereological relations, such as functional decomposition, spatial configuration, topology etc.. Subsumption between classes is valid only if their unity criteria are compatible: whether a class carries unity, anti-unity or no unity is inherited to its subclasses.

Formal Evaluation

The ONTOCLEAN methodology provides a formal framework for evaluating ontologies. It is instrumental in the formal elicitation of the ontological decisions underlying subsumption relations. Annotating classes with meta properties allows for automatic evaluation of the taxonomic relations in the ontology, i.e. if a class C subsumes C′ then:

- If C is *anti-rigid*, then C′ is anti-rigid: all students, be they lazy or overzealous can graduate without ceasing to exist.
- If C carries some identity criterion, then C′ must carry the same criterion: if organisms can be uniquely identified by their DNA, then any homo sapiens, canis familiaris, limax maximus and amoeba proteus can be as well.
- If C carries some unity criterion, then C′ must carry the same unity criterion: if a table consists of physical parts that enable its function in providing a level surface on which objects can be placed, then all tables do.
- If C has *anti-unity*, then C′ must also have anti-unity: if water does not have parts, then potable water doesn't either.

Meta properties are not just helpful when evaluating an ontology or constructing a new ontology. They can be used to put additional constraints on the validity of ontology reuse, provided that both the new and the reused ontology are annotated. However, the annotations are meta statements over axioms in the ontology and are not part of their semantics. Though formal, they cannot be used in the same way as the definition for conservative extensions to solve the ontology interaction problem (definitions 5.4.6 and 5.4.5, respectively). Rector (2003) recognised the two approaches as complementary parts of a two-step normalisation: ontological and implementation normalisation.

A limitation of the ONTOCLEAN methodology is that the distinctions it advocates, though inspired by philosophy, are an abstraction of established practices in knowledge engineering. For instance, the notion of anti-rigidity reflects a meta-theory of the distinction between roles and role-players (Steimann, 2000; Masolo et al., 2004), properties that are always *attributed* to some entity (cf. Section 6.3.2, Section 7.3.2). They reflect things an entity can *have* or *play*. Similarly, the notion of unity stands in direct correspondence to the distinction between objects and the substances they are made of.

Guarino and Welty are careful to avoid presenting ONTOCLEAN as a traditional methodology that provide guidelines for *taking* modelling decisions, rather the meta-properties are purely metadata that *signal* a particular ontological position with respect to some category. Though making these meta-properties explicit helps to communicate design decisions, the rationale for making these decisions is lost. In fact, because these meta-properties are derived from more concrete distinctions it begs the question whether the ONTOCLEAN approach gives more solid footing for an ontology engineer than a concrete guideline would.

Furthermore, that meta properties are enforced down the subsumption hierarchy holds for other properties as well, and is provided 'for free' by any serious knowledge representation language. That an ontology engineer needs to take this inheritance into account is indeed a source of many modelling errors, but is not particular to the meta-properties of the OntoClean approach.

5.5.2 Frameworks and Ontologies

The OntoClean methodology sprouted from a need to provide a means to evaluate the correctness of taxonomic relations, but is silent where it concerns the appropriateness of categories included in the ontology. In ontology representation languages such as OWL, the distinction between 'ontology' and actual situations coincides with the contrast between terminological knowledge and assertional knowledge: classes in the TBox and individuals in the ABox. However, this distinction is not concise enough to separate proper ontological categories from other, general terminological knowledge. In fact, not all concepts are suited for inclusion in an ontology, and in several papers we advocate a distinction between ontologies and so-called *frameworks* (Breuker and Hoekstra, 2004c; Breuker et al., 2007; Hoekstra et al., 2007, 2008).

As a rule, terminological knowledge is generic knowledge while assertional knowledge describes some state of affairs. These states can be generalised to patterns typical to particular kinds of situations. To be sure, if some pattern reoccurs and has a justifiable structure, it might evidently pay to store this structure as generic description. For instance, it may capture a predictable course of events. The combination of the situations and events related to eating in a restaurant is a typical example, and served in the Seventies to illustrate the notion of knowledge represented by scripts or 'frames' (Schank and Abelson, 1977; Minsky, 1975, respectively). A representation of this kind of generic knowledge, which is indeed terminological, is not an ontology.

Frameworks have a different structure than ontologies and capture systematic co-occurrence of the structural *relations* something has with other things. They describe such things as how activities are causally or intentionally related, or how objects are spatially and functionally configured: frameworks are primarily defined through *unity criteria*. Ontologies, on the other hand, define what things *are* in and of themselves and emphasise the essential, intrinsic properties of entities. Arguably, at a formal level the two are indistinguishable: every class in OWL is defined in relation to other classes, it cannot be otherwise. The distinction between frameworks and ontologies is therefore conceptual and does not coincide with the attributes of representation formalisms. For this reason, to keep an ontology free from contextual knowledge, we need to distinguish between those relations that make what a concept *is* and relations that place a concept in a particular frame of reference. For sure, the ontological definition of a concept is dependent on its context within the structure of the ontology, but it should be impartial to the context of its *instances*.

Take for example a *hammer*, its composition of head and shaft is not by accident: it is this particular combination that allows the hammer to be used 'as a hammer'. However, the mereological relation between the hammer as object, and its composites is not part of its ontological definition. Many different kinds of hammers exist, e.g. sledgehammers, mallets, conveniently shaped stones etc., each of which differs ever so slightly in the nature of its composites

and the relations between them. But they are all hammers. It is therefore only the function of the hammer as an instrument that defines what a hammer *is*, its mereological properties are merely circumstantial evidence. Searle (1995) stressed that the hammer-as-such cannot be substituted with the hammer-as-object; whether some object is a hammer is a social fact and depends on an attribution of the hammer function given a context of use. This attribution pinpoints exactly the conceptual difference between ontologies and frameworks. Where frameworks may incorporate the context in which this attribution holds by default and thus (over) generalise the typical physical features of a hammer-as-object to the hammer-as-such, ontologies should maintain an explicit distinction between the two.

From a methodological point of view, this allows us to introduce a rule of thumb: a combination of concepts and relations, as in e.g. a class restriction, is only to be considered part of an ontology when this particular combination is either systematic and independent of context, or when it makes the context explicit. A possible second consideration is inspired by the limitations imposed on admissible structures by the knowledge representation formalism. Of course this is a rather practical consideration, but nonetheless useful when considering the widespread use of OWL. If the primary task of ontology is to describe what things are, then a representation formalism specifically tailored to classification can be a fair benchmark for determining whether the kind of knowledge you want to express should be considered part of ontology or not. Given the limited expressiveness of OWL DL, especially because of its tree-model property (Motik et al., 2005), many frameworks are more easily represented using a rule formalism as they often require the use of variables. In fact, as we show in Hoekstra and Breuker (2008) and Section 7.2, OWL can be used to represent non-tree like structures commonly found in frameworks but only in a very round-about and non-intuitive manner. Furthermore, *epistemological frameworks* may define epistemic roles which can only be applied by reasoning architectures that go beyond the services provided by OWL DL reasoners (e.g. when they require meta-level reasoning). The limitations of OWL thus indicate a correlation between the conceptual distinction and representation formalisms. However, the two perspectives should not be confounded. Frameworks belong to the T-Box of any knowledge representation system, independent of whether it is based on a DL formalism or not.

We can distinguish three kinds of frameworks:

Situational frameworks

Situational frameworks are most characteristic for the notion of framework in general, because of the strong emphasis on context and teleology they share with frames and scripts (Minsky, 1975; Schank and Abelson, 1977). They are the stereotypical structures we use as plans for achieving our goals given some recurrent context, such as making coffee. These plans are not necessarily private to a single agent, and may involve transactions in which more than one actor participates. For instance, the definition of Eating-in-a-restaurant[24] is a structure consisting mainly of dependencies between the actions of clients and service personnel.

[24] In the following all concepts will start with a capital letter, properties and relations will not

Another notable characteristic of situational frameworks is that they are not subclasses of the goal directed actions they describe. For instance, Eating-in-a-restaurant is not a *natural* sub-class of Eating but rather refers to a typical model of this action in the situational context of a restaurant. Furthermore, it usually does not make sense to define subclass relations between situational frameworks themselves. Although we can easily envisage a proliferation of all possible contexts of eating – Eating-at-home, Eating-with-family, etc. — but does eating in a French restaurant fundamentally differ from eating in a restaurant in general? (Bodenreider et al., 2004; Breuker and Hoekstra, 2004a).

From a legal perspective, situational frameworks can be imposed on actual situations through articles of procedural ('formal') law. Although the *stereotypical* plans given in by custom, and the *prescribed* plans of law differ in their justification – rationality vs. authority – their representation is largely analogous. Similarly, legal norms combine generic situation descriptions with some specific state or action, where the description is qualified by a deontic term. For instance, the norm that "vehicles should keep to the right of the road" states that the situation in which a vehicle keeps to the right is obliged.

In short, situational frameworks are a fundamental part of the way in which we makes sense of the world, and play a prominent role in problem solving.

Mereological frameworks

Most entities, and objects and processes in particular, can be decomposed into several parts: they are *composites*. As we have seen in the hammer example, it can be tempting to incorporate a mereological view in the definition of a concept. A typical example is the definition of Car as having at least three, and usually four wheels, and at least one motor. However, a full *structural* description of a concept's parts and connections goes beyond what it *essentially* is. Cars are designed with a specific *function* in mind, and although there are certainly some constraints on its physical properties relevant to its definition, these are limited to those constraints actually necessary for fulfilling that function. Concept descriptions that do iterate over all or most parts of the concept are *mereological frameworks*, and can appear under a large diversity of names: structural models, configurations, designs, etc. Mereological frameworks play a major role in qualitative reasoning systems (see e.g. Davis (1984); Hamscher et al. (1992)).

Arguably, the line between the mereological framework and ontological definition of a term is sometimes very thin. For instance, if we want to describe a bicycle as distinct from a tricycle, it is necessary to use the cardinality of the wheels as defining properties as these are *central* to the nature of the bicycle. On the other hand, the number of branches a tree might have hardly provides any information as to what a tree *is*.

Epistemological frameworks

Inference in knowledge based reasoning does not happen in isolation. It is part of a larger structure of interdependencies between steps in a reasoning process: arriving at new information, given some initial situation. Epistemology is the study of how valid knowledge can be obtained from facts. Generic description

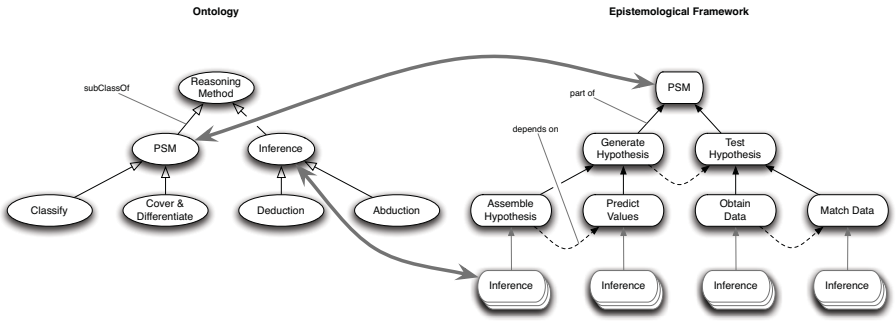

Figure 5.6: Ontology of reasoning terms vs. a framework for reasoning (Breuker and Hoekstra, 2004a)

of reasoning processes are thus *epistemological frameworks* that focus on the epistemological status of knowledge: the *use* of knowledge in reasoning, e.g. as hypothesis or conclusion (Breuker and Hoekstra, 2004c).

What sets the epistemological framework apart from mereological frameworks is its characteristic specification of dependencies between the distinct steps in a reasoning methodology. A typical example are the problem solving methods (PSM) discussed in Section 2.4.1 that are found in libraries of problem solving components (Breuker and Van De Velde, 1994; Motta, 1999; Schreiber et al., 2000). For instance, a problem solving method is not just a break-down of the mereological structure of some problem, but specifies control over the inferences it contains, such as the assessment of success or failure in attaining a sub goal. They invariably have at least two components: some method for selecting or generating potential solutions (hypotheses), and a method for testing whether the solutions hold. Whether a solution holds depends on whether it satisfies all requirements, or whether it corresponds with, *explains*, empirical data.

Figure 5.6 shows how an ontology of problem solving differs in perspective from a problem solving method described as epistemological framework. Although the concepts *inference* and *PSM* occur in both, the structure in which they are placed is decidedly different, as is also demonstrated by their sub categories. The epistemological framework has a mereological structure combined with sequential dependencies between its parts, the ontology is a subsumption hierarchy. To give a further illustration of the difference, consider the following action a:

> "Colonel Mustard kills Dr. Black in the conservatory, with the candlestick."

In a game of Cluedo a will typically occur first as *hypothesis*, before it may or may not be accepted as *conclusion* (Colonel Mustard actually *did* kill Dr. Black). Being a hypothesis is therefore not essential to a, and in fact it never is: the hypothesis class is anti-rigid. Conversely the content of a hypothesis is not ontologically relevant to the concept itself; that a is a different hypothesis from say "The candlestick has a huge dent in it" is only particular to the two individual hypotheses, but does not change what a hypothesis *is*. Similar to the Eating-in-

a-restaurant situation, the relevance of an α-hypothesis class is dependent on the epistemological context of a particular problem: whodunnit?

Ontology versus Epistemology In general, the categories of epistemological frameworks can be defined in ontologies, provided that their definition is ontological: in Chapter 2 we have seen ample reason for keeping a strict separation between task dependent knowledge and domain knowledge, cf. the interaction problem of Bylander and Chandrasekaran (1987). Nonetheless, Bodenreider et al. (2004) discuss several other cases where (medical) ontologies are interspersed with epistemological categories of a different kind:

- Terms containing *classification criteria*, i.e. categories distinguished according to the way information is gathered to classify a particular instance as belonging to the category, e.g. *Murderer, convicted by jury trial* versus *Murderer, convicted by bench trial*. Although such information may be useful in justifying the conclusions of a knowledge based system, it does not constitute an ontological category.
- Terms reflecting *detectability*, *modality* and *vagueness*, e.g. *asymptomatic* diseases, that are detected without the patient displaying its symptoms. This information does not constitute a new ontological category, i.e. the treatment of an asymptomatic disease is the same as for symptomatic diseases.
- Terms created in order to obtain a *complete partition* of a given domain, such as *car*, *lorry*, *bicycle*, *other vehicles*. Often such bin-concepts are introduced because of a database perspective, where all categories need to be explicitly named.
- Terms reflecting mere *fiat boundaries*, e.g. a person of *average height* will have a different height in Asia than in Europe.

From a philosophical perspective the practice of having epistemologically loaded categories is problematic as these are decidedly non-ontological and do not correspond to categories of entities in reality. Knowledge representation ontologies are less scrupulous with regard to the ontological status of their definitions, and often some concessions will need to be made to facilitate the implementation of the ontology in a practical implementation (e.g. in application ontologies). However, epistemological promiscuity of categories is problematic as they are defined within the context of some shared epistemological framework or task. For instance, a category of asymptomatic diseases is only of relevance in the context of medical diagnosis, not in their treatment. And different trial procedures are mainly relevant when comparing different legal systems, but not when establishing a sentence.

Alles Vergängliche ist nur ein Gleichnis

J.W. Goethe, Faust II, 1832

5.6 Design Patterns

Most of the requirements and principles of the preceding sections pertain to what van Heijst et al. (1997) called *knowledge modelling* (Section 2.4.1): they are largely conceptual and independent of the language in which the ontology is eventually specified. However, a formally specified ontology is a *design model*, where the design is subject to *design principles* that guide the expression of the conceptual model in a formal language. As pointed out in Section 4.2, the language of choice for knowledge engineering ontologies that are to be shared on the web is the tailor made description logics formalism of OWL DL.

Knowledge modelling and design are not independent. Firstly, the choice of a representation language, and in particular its expressiveness, determines which (kinds of) ontologies can be reused. And secondly, safe reuse cannot be defined without reference to the language in which either ontology is represented. In both cases, the expressiveness of the ontology representation language is key, and in fact has a significant impact on what parts of the ontology as conceptual model can be formalised. In particular, the trade-off between expressiveness on the one hand and decidability and computational efficiency on the other played an important role in the specification of OWL DL (Levesque and Brachman, 1987, and Section 4.2). This technical trade-off is reflected by an ontological trade-off:

- not all ontologically relevant features can always be represented in OWL DL, and
- some features can be represented in multiple ways, each with its own benefits.

The encoding of an ontology is thus not just the result of several ontological choices, but also of design decisions. The requirements we reviewed so far do not directly support this process. For sure, it is the reuse of pre existing ontologies that lifts some of this burden. Not only does it provide off-the-shelve concepts and properties, but these convey insight in the decisions underlying the design of the ontology as well. The parallel with ontological choices has not gone unnoticed, and akin to the ontological principles of the previous section, the design decisions underlying several ontologies can be extracted as ontology *design patterns*:

Definition 5.6.1 (Design Pattern) *A* design pattern *is an archetypical solution to a design problem given a certain context.*

5.6.1 Patterns and Metaphors

Ontological definitions of concepts in a formal language do not differ much from descriptions of terms in natural language. In both cases an expression is constructed by combining symbols according to grammatical rules. The role of design patterns in ontologies can be made clear by analogy to the use of grammatical patterns in meaning construction. In fact, the way in which we combine words in linguistic expressions hints at the existence of several fundamental concepts in thought, and follows basic, cognitive rules (Pinker, 2007). One of the examples Pinker gives to support this hypothesis is the atypical behaviour of some verbs in the dative form. While most transitive verbs can be used both in a propositional dative form (subject-verb-'to'-recipient) and a double-object form (subject-verb-recipient-thing), this cannot be extracted to a general rule. For instance, whereas we can say:

- 'Give a muffin to a moose'
- 'Give a moose a muffin'

we cannot say:

- 'Biff drove the car to Chicago'
- 'Biff drove Chicago the car'

or, alternatively:

- 'Sal gave Jason a headache'
- 'Sal gave a headache to Jason'

Idiosyncrasies such as this indicate that the two constructions are not synonymous, but in fact follow differing underlying patterns. Whereas the propositional dative matches the pattern "cause to go", as in '*cause a muffin to go to a moose*', the double-object dative matches "cause to have", as in '*cause a moose to have a muffin*'. A plethora of other constructions such as these exists. For example, to indicate a distinction between direct and indirect causation as in '*dimming the lights*' when sliding a switch and '*making the lights dim*' when turning on the toaster. It is these patterns that are used to construct metaphors such as the *container* metaphor of Section 5.6.3 or a *conduit* metaphor where ideas are things, knowing is having, communicating is sending and language is the package:

> We *gather* our ideas *put* them *into* words, and if our verbiage is not *empty* or *hollow*, we might *get* these ideas *across* to a listener, who can *unpack* our words to *extract* their *content*.
>
> (Pinker, 2007, p.60)

According to Pinker, the basic concepts in a language of thought correspond to the kinds of concepts that fit the slots of grammatical constructions. The grammar rules of language reflect the structure of our conceptualisation of the world around us. It shows the restrictions on the ways in which we

can combine concepts to create meaningful categories. This is not the place to discuss whether or not Pinker's basic concepts are ontologically relevant (cf. Chapter 6), rather what is important here is that the position of some term in a sentence is indicative of its meaning, and the general category it belongs to.

We have seen that linguistic expressions follow patterns that can be re-applied to new circumstances to create new meaning. These patterns are design patterns, and depending on *which* pattern we apply, we create a different meaning or shift emphasis. However, not every pattern is applicable to just any term. In Pinker's example, the applicability of some grammatical pattern to a term is determined by an (in)compatibility between that which a pattern is meant to express about it, and the meaning of the word. As we will see, design patterns in knowledge representation and ontology development pose similar restrictions

5.6.2 Patterns in Ontology Engineering

Presutti et al. (2008) give a comprehensive overview of design patterns in ontology engineering and discuss a wide range of patterns. Though related to ontology engineering, not all of these capture design decisions. For instance, the definition is used to specify presentation patterns (naming conventions), documentation best practices (annotating classes), patterns for ontology decomposition, mappings between ontology vocabularies (e.g. SKOS to OWL), normalisation macros (asserting inferred information), and even grammar rules for translating natural language sentences into OWL axioms.

Their definition of ontology design pattern is influenced by its epistemological role as data structure in the specification of design patterns in an online library. For this application, it is useful to view a engineering design pattern as reification of the relation between a design schema (and its elements), and its possible implementations (and their elements):

Definition 5.6.2 (Ontology Engineering Design Pattern (Presutti et al., 2008)) *An* ontology design pattern *is a modelling solution to solve a recurrent ontology design problem. It is an* information object *that expresses a* design pattern schema *or* skin*. A schema can only be satisfied by a* design solution*, which is a particular combination of ontology elements that fulfil certain* element roles *defined by the design pattern.*

Using this definition, patterns and their solutions can be properly indexed and documented. Furthermore, the definition is cast in terms of the DOLCE Ultra light ontology and the Descriptions & Situations extension of Gangemi and Mika (2003), which gives it a grounding in a larger foundational ontology. Nonetheless, the fact that the definition presents a design pattern as a reification obscures the reason *why* a solution 'satisfies' a schema. Especially since the ontological relevance of an individual design pattern relies on this relation, a proper definition should specify how this relevance can be assessed. This assessment should not rely on an explicitly asserted relation with concrete implementations. It seems that ontologically relevant design patterns are *themselves* the generic 'pieces' of a formal specification that can be applied, or implemented by an ontology.

5.6.3 Knowledge Patterns

Design patterns are not just generic descriptions that can be inherited over a subsumption hierarchy (Clark et al., 2000). The application of a design pattern in a concrete case is a source of inference when it applies to multiple aspects of the case. Clark et al. give a logic programming example of a container pattern that is used to express the capacity of a computer: "the computer contains memory" Arguably the term 'capacity' is ambiguous and may refer to a variety of measurable attributes of the computer, e.g. internal memory, hard disk space, or the number of expansion slots. Querying a DL knowledge base that applies the pattern in a straightforward way will return all possible capacities.

One solution is to *parameterise* the pattern by requiring the capacity to be of a certain type. Clark et al. reject this solution on the grounds that parsimony may be even further compromised when more distinctions are necessary, such as between physical and metaphysical containment ("the computer contains information"). In short, to ensure applicability to a broad range of cases a design pattern will need to explicitly include all possible dimensions along which its cases may differ. Such patterns will be overly complex and include a domain dependent bias: they cannot be applied freely to new cases.

The conclusion is that although it may seem that a subsumption relation holds between the container and a computer, this is incorrect. Rather, a computer can be *modelled as* a container: we apply a container metaphor. Clark et al. (2000) define a *knowledge pattern* as a theory whose axioms are not part of the knowledge base that implements it. The implementation of such a pattern occurs by importing its axioms, and mapping the symbols of the implementation to symbols in the pattern's signature. This mapping is given by a *morphism* that maps every non-logical symbol in the pattern to a corresponding symbol in the knowledge base. For instance, we can define Free_Space as that capacity of a Container that is not occupied:

$$\begin{aligned} \mathsf{Free\_Space} &\sqsubseteq \mathsf{Capacity} \sqcap \mathsf{capacity\_of}\ \textbf{some}\ \mathsf{Container} \sqcap \textbf{not}\ \mathsf{Occupied} \\ \mathsf{Container} &\sqsubseteq \mathsf{free\_space}\ \textbf{some}\ \mathsf{Free\_Space} \\ \mathsf{free\_space} &\sqsubseteq \mathsf{capacity\_of}^{-} \end{aligned}$$

This allows us to query the value of the free_space property to retrieve the free space of any container. By applying the morphism m:

$$\begin{aligned} m = \{&(\mathsf{Container}, \mathsf{Computer}), (\mathsf{Free\_Space}, \mathsf{Available\_RAM}), \\ &(\mathsf{Capacity}, \mathsf{RAM\_Size}), (\mathsf{Occupied}, \mathsf{Used\_RAM}), \\ &(\mathsf{free\_space}, \mathsf{available\_ram})\} \end{aligned}$$

we can construct the analogous definitions for Available_RAM and Computer:

$$\begin{aligned} \mathsf{Available\_RAM} &\sqsubseteq \mathsf{RAM\_Size} \sqcap \mathsf{capacity\_of}\ \textbf{some}\ \mathsf{Computer} \sqcap \textbf{not}\ \mathsf{Used\_Ram} \\ \mathsf{Computer} &\sqsubseteq \mathsf{available\_ram}\ \textbf{some}\ \mathsf{Available\_RAM} \\ \mathsf{available\_ram} &\sqsubseteq \mathsf{capacity\_of}^{-} \end{aligned}$$

Similar morphisms can be defined that map e.g. Free_Space onto Free_Slots or Free_Harddisk_Space etc.

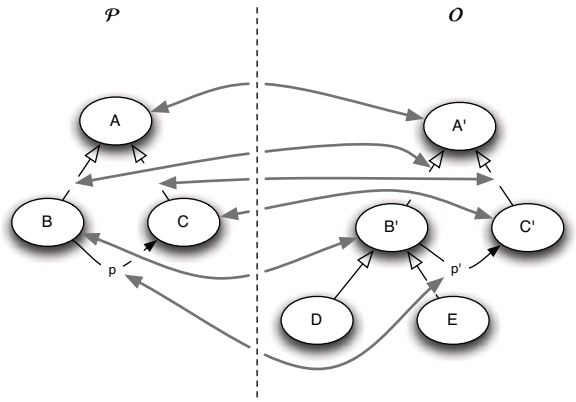

Figure 5.7: The mapping between ontology design pattern $\mathcal{P}$ and its implementation in ontology $\mathcal{O}$

The definition of knowledge pattern circumvents the limitations of pattern implementation based solely on subsumption, and introduces a weaker mapping function to connect patterns to their implementation. Nonetheless, knowledge patterns are rather restrictive in that both the pattern and the mapping are required to be external to the knowledge base.

5.6.4 Ontology Design Patterns

Viewing ontology design patterns as knowledge patterns allows for a formal evaluation of pattern implementation in ontologies. In this section I formalise and extend the most salient patterns of Presutti et al. (2008), and scrutinise them with respect to the reuse criteria discussed in Section 5.4.2. I define an ontology design pattern as a knowledge pattern that captures an ontological theory and can be implemented in other ontologies:

Definition 5.6.3 (Ontology Design Pattern) *An* ontology design pattern *is an ontology $\mathcal{P}$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff a mapping function M specified by a* signature morphism *exists between the signatures $\mathsf{Sig}(\mathcal{P})$ and $\mathsf{Sig}(\mathcal{O})$ that for all axioms $\alpha \in \mathcal{P}$ it holds that $\mathcal{O} \models M(\mathsf{Sig}(\alpha), \alpha)$, where $M(\mathsf{Sig}(\alpha), \alpha)$ is the set of axioms produced by applying M to α and its signature.*

Figure 5.7 shows how a design pattern $\mathcal{P}$ is mapped onto ontology $\mathcal{O}$. Note however, that like knowledge patterns, this definition does not strictly require a bijective mapping between pattern and implementation. A symbol from the signature of $\mathcal{O}$ may occur an arbitrary number of times in the mapping M, and not all symbols from $\mathcal{O}$ must be mapped onto the signature of $\mathcal{P}$ (although all those from $\mathcal{P}$ must). Furthermore, it can be useful to weaken the definition of knowledge pattern to allow the mapping to be defined as axioms in $\mathcal{O}$. This way, the implementation relation does neither require nor preclude the use of subsumption relations to map the pattern ontology $\mathcal{P}$ to relevant parts of $\mathcal{O}$. Although this leaves room for some of the ambiguities Clark et al.'s definition is meant to circumvent, it allows for a straightforward implementation of the pattern in cases where the ambiguity does not hurt.

Presutti et al. (2008) distinguish *content* patterns and *logical* design patterns. Content patterns are in fact small ontologies, the reusability of which is advanced by explicit documentation of their design rationale and best practices. Gangemi (2005); Presutti et al. (2008) propose the development of a library of such content patterns that can be used as building blocks in ontology design. Logical design patterns are specified at a meta-level and implemented by *instantiation*. Axioms in the implementation are typed according to the meta-categories defined by the pattern.

Content patterns are defined as restriction on the knowledge pattern of Clark et al. (2000), on which definition 5.6.3 is based. Rather than a free mapping relation between pattern and ontology, an implementation of a content pattern should preserve *downward taxonomic ordering*. The signature of the content pattern is mapped only to symbols that are subsumed by symbols of the pattern. Presutti et al. (2008) do not give formal definitions of pattern implementation, but the gist of their proposal can be captured by an adaptation of definition 5.6.3:

Definition 5.6.4 (Content Pattern) *A* content ontology design pattern *is an ontology $\mathcal{P}_c$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff $\mathcal{P}_c$ is implemented as ontology design pattern in $\mathcal{O}$, and if $\mathcal{P}_c$ were in its axiom closure (i.e. $\mathcal{P}_c \subseteq \mathcal{O}_\cup$) then for all axioms $\alpha \in \mathcal{P}_c$ it would additionally hold that $M(\mathit{Sig}(\alpha), \alpha) \sqsubseteq \alpha$ in $\mathcal{O}$.*

In other words, the axioms in $\mathcal{O}$ returned by the mapping function M applied to α and its signature must be subsumed by α.

If an ontology design pattern is itself an ontology, this raises the important question what its ontological status is. For instance, should a naive implementation of a pattern were $\mathcal{P}$ is imported and extended by $\mathcal{O}$ using standard OWL constructs meet the same safety requirements as normal ontology reuse? It is clear that the additional restriction on the implementation of a pattern posed by definition 5.6.4 makes that, though actual reuse of the pattern *as ontology* is not required, it still must be a conservative extension (definition 5.4.6). Consequently we can apply the criteria of a.o. Ghilardi et al. (2006); Cuenca Grau et al. (2007) to the union $\mathcal{P}_c \cup \mathcal{O}_\cup$ to determine whether $\mathcal{O}$ is a *safe implementation* of content pattern $\mathcal{P}_c$.

Because of their meta level character, logical design patterns are a different breed and have a more structural character. In Presutti et al. (2008), logical patterns are logical meta theories. To illustrate this, consider the top level categories of OWL itself. These are the language constructs that enable the specification of *any* valid OWL ontology. Logical patterns work in the same way; they define the meta classes and reified relations that specify valid implementations of the pattern. To give a formal definition:

Definition 5.6.5 (Logical Pattern) *A* logical pattern *is an ontology $\mathcal{P}_l$ that is said to be* implemented *in an ontology $\mathcal{O}$ iff there exists a set of axioms $\mathcal{Q}$ such that $\mathcal{Q} \subseteq \mathcal{O}$ and it holds that $\mathcal{P}_l \models \mathcal{Q}$.*

The part of an ontology that implements a logical pattern should be a valid model of the pattern. Logical patterns expressed in OWL cannot be enforced using a description logics classifier; they are limited to OWL Full (cf. Section 3.3.2). In other words, the correctness and safeness of a logical pattern implementation cannot be ensured. Furthermore, implementing meta level

statements is not structure preserving; a valid implementation does not need to have the same structure as the pattern it implements. The reason is that where axioms in content patterns are imported, copied or extended, axioms in logical patterns are *instantiated*.

Design patterns are not always explicitly specified as a formal theory. They are rather *described* as step-by-step procedures larded with stereotypical examples. For instance, the procedures for specifying value partitions (Rector et al., 2004; Rector, 2005), imitating N-ary relations (Noy and Rector, 2006), classes as values (Noy, 2005), transitive propagation (Seidenberg and Rector, 2006), sequences (Drummond et al., 2006) and structured objects (Motik et al. (2007a); Hoekstra and Breuker (2008) and Chapter 7). Largely, these procedures are not intended as ontological commitment – they do not refer to entities in reality – but are rather meant to describe how certain structures can be approximated that cannot be directly expressed in OWL.

If we take definition 5.6.5 at face value, the patterns described by such procedures would only count as logical pattern when formalised as a meta theory. However, given the limitations of meta theories in OWL, and the generality of examples in pattern descriptions, the utility of a strict logical pattern does not necessarily transcend that of less formal patterns. What sets the patterns described above apart from content patterns, is that the mapping between pattern and implementation is not subject to ontological restrictions. It does not assume an ontological relation (e.g. subsumption) between the symbols in either signature, but merely ensures the transposition of the *structure* of the pattern to categories in the implementation. We can therefore introduce a third type of pattern, not described in Presutti et al. (2008), which is neither a *content pattern* nor a *logical pattern*.:

Definition 5.6.6 (Structure Pattern) *A* structure pattern *is an ontology design pattern for which the mapping function M is* bijective*, and no semantic requirements hold as regards the relation between elements in the pattern and its implementation.*

Because of the allowed ontological disconnect between structure patterns and implementing ontologies, their implementation does not explicitly involve an ontological commitment. However, it may incorporate an *epistemological* commitment of the type identified by Bodenreider et al. (2004). Structure pattern implementations signal an intended interpretation for that part of the ontology. For instance, an OWL implementation of the N-ary relation pattern of Noy and Rector (2006) signals that even though it *is not* an N-ary relation according to the OWL semantics, it should be interpreted as such. This works in exactly the same way that the presence of OWL constructs signal the applicability of OWL semantics.

The epistemological nature of structure patterns is not a given, and can be said to decrease when their intended interpretation is more closely approximated by the standard semantics of their implementation. For example, where the intended and actual semantics of the patterns in Rector (2005); Seidenberg and Rector (2006) coincide, and those of Drummond et al. (2006); Hoekstra and Breuker (2008) are close approximations, the patterns of Noy (2005); Noy and Rector (2006) make more conservative claims.

Although it is certainly true that some of these patterns are motivated mainly by epistemological objectives, and do not have explicit ontological commitments, they do convey a certain ontological message. Design patterns may

transpose the stereotypical structure of ontological categories to new domains. In other words, the implementation of a structure pattern can still constitute a metaphor, but this depends on the strength of the ontological relation between a pattern and its implementation. Rather than complete downward taxonomic ordering as in content patterns, metaphoric use of a structure pattern depends on the reuse of *relations* between categories:

Definition 5.6.7 (Metaphoric Use) *The* metaphoric use *of a structure pattern $\mathcal{P}_s$ in ontology $\mathcal{O}$ requires that $\mathcal{O}$ implements $\mathcal{P}_s$, and that if $\mathcal{P}_s$ is in the axiom closure of $\mathcal{O}$ (i.e. $\mathcal{P}_s \subseteq \mathcal{O}_\cup$) then for all axioms $\alpha \in \mathcal{P}_s$ it would additionally hold that for its* properties *$p_s \in \mathsf{Sig}(\alpha)$ and each corresponding property $p \in \mathsf{Sig}(M(\mathsf{Sig}(\alpha), \alpha))$ it holds that that $p \sqsubseteq p_s$ in $\mathcal{O}$.*

Metaphoric use of a structure pattern thus requires that the properties of an implementing ontology are at least (conceptually) subproperties of corresponding properties in the pattern. Where the implementation of content patterns requires a subsumption relation between *all* elements of the signature. To illustrate the difference, the conduit metaphor from Pinker (2007) in Section 5.6.1 cannot be implemented as content pattern, but can be a metaphoric use of a structure pattern. For instance, the former requires words to literally *be* containers for ideas, and the latter merely signals a connotation. The relational character of metaphoric use corresponds to the role of verbs in natural language metaphors (Pinker, 2007), and the prototypical representation of verbs as properties (Brachman et al., 1991).

5.7 Discussion

In this chapter, I discussed general methodological guidelines for ontology development, emphasised the middle-out approach for constructing an ontology and gave a characterisation of different types of ontologies available for reuse. Section 5.4.2 described the technical restrictions we need to impose to guarantee safe reuse, without compromising the ontological commitments of the reused ontology. In Section 5.5, these technical considerations are augmented with a means to make explicit part of these ontological commitments, that can be used to ensure conceptually sound extensions. The notion of *framework*, and in particular the epistemological framework, allows us to separate ontological categories from other terminological knowledge. Design patterns allow us to express partial ontological theories that can be transposed and used as metaphors.

Most, if not all of these considerations are given by the ideal of ontology sharing and reuse. Indeed, all theoretical and philosophical notions aside, someone who uses the DL semantics of OWL to build just terminological knowledge bases may not heed these requirements. Nonetheless, once a knowledge base is available on the web it will inevitably be evaluated on its merits as *reusable knowledge source*. Ontologies are not 'just' terminological knowledge bases, a view that is emphasised by the wide range of guidelines, requirements and resources outlined in this chapter. Ontology development remains an art, and no a priori guidelines exist that can ensure the quality of the result. Nonetheless, the emphasis on an ontological perspective of Section 5.5, the reuse

of existing ontologies described in Section 5.4 and the restrictions on reuse and design patterns of Section 5.6 and Section 5.4.2 are important constraints on ontology development. Although these constraints are not sufficient, they are the *necessary* conditions for quality ontologies, and perhaps give a better flavour of what an ontology *is* than any definition – let alone Gruber's – could.

The next chapter shows how the principles outlined in this chapter are applied in the construction of a common-sense based core ontology for law: the LKIF Core ontology. In Chapter 7 several important design patterns adopted in this ontology are highlighted and discussed in more detail.