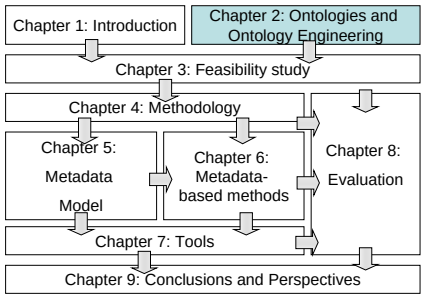
2 Ontologies and Ontology Engineering



.

In this chapter we introduce the fundamental principles behind ontologies, ontology engineering and the Semantic Web. After summarizing the mainstream understanding of the former and some of the most relevant methodologies, methods and tools aiming at developing and managing ontologies in Sections 2.1 and 2.2, respectively, we describe typical application settings which are proved to benefit from the usage of declarative domain knowledge conceptualizations in form of ontologies in Section 2.3. We close the chapter with a short summary (Section 2.4)

2.1 Ontologies

The term *"ontology"* has been introduced to computer science as a means to formalize the kinds of things that can be talked about in a system or a context. With a long-standing tradition in philosophy, where *"**O**ntology"* denotes *"the study of being or existence"* [30, 98], ontologies provide knowledge engineering and artificial intelligence support for modelling some domain of the world in terms of labeled concepts, attributes and relationships, usually classified in specialization/generalization hierarchies.[1] With applications in fields such as knowledge management, information retrieval, natural language processing, eCommerce, information integration or the emerging Semantic Web, ontologies are part of a new approach to building intelligent information systems [60]: they are intended to provide knowledge engineers with *reusable pieces of declarative knowledge*, which can be—together with *problem-solving methods* and *reasoning services*—easily assembled to high-quality and cost-effective systems [154].

According to this idea, ontologies are understood as means to *share and reuse* declarative knowledge. They enable a novel knowledge-based systems development strategy, in which application or domain knowledge is strictly separated from software implementations and can be thus *efficiently* reused across heterogeneous software platforms [86].

The emergence of the Semantic Web has marked an important stage in the evolution of ontologies. Primarily introduced by Tim Berners Lee [12], the originator of the World Wide Web, the idea of providing the current Web with a computer-processable knowledge infrastructure in addition to its actual, semi-formal and human-understandable content foresees the

[1] Guarino and Giaretta propose to use different notations for the philosophical and the knowledge engineering views upon ontologies. According to this distinction the philosophical sense is denoted by the capitalized word *"**O**ntology"*, while the second one is written without capitals [87] and may be used in plural form.

usage of knowledge components which can be easily integrated into and exchanged among arbitrary software environments *in an operationalized manner*. In this context the knowledge components, i.e. the ontologies, are formalized using Web-suitable, but in the same time semantically unambiguous representation languages, are pervasively accessible and can be (at least theoretically) shared and reused across the World Wide Web.

2.1.1 Definitions

While the philosophical foundations of ontologies as systematic study of being are widely acknowledged, the clear boundaries of this concept within and across research communities in computer science are not fully agreed yet. As a consequence we encounter several, though complementary perspectives on ontologies, which associate this concept with various *scopes* and *degrees of formality*.

The most popular definition of the term originates from Gruber [84]: an ontology is an *"explicit, formal specification of a shared conceptualization"*. In terms of this definition an ontology is characterized by the subsequent features [209]:

- it is a *formal* and *explicit*: this means that the ontology is represented using a formal language i.e. a language with a machine-understandable semantics and that it uses types of primitives (e.g., concepts, axioms) which are explicitly defined.
- it is *shared*: the ontology mirrors a common understanding of the modelled domain, being the result of a consensus achieved within a (potential) community of ontology users.
- it *specifies a conceptualization*: the ontology is used to model some domain of the world. This implies that the ontology is responsible for a specific view upon the corresponding domain, reflected in certain simplifications, abstractions, omissions or other modelling decisions.

Alternative ways to define ontologies usually highlight only a subset of the aforementioned aspects. A logics-centered view is held by Guarino [86, 87]: an ontology is a *"logical theory which gives an explicit, partial account of a conceptualization"*. This definition, though sharing several commonalities with the one from Gruber [84], reduces ontologies to *logical structures*, thus excluding conceptual models which do not commit to a logically precise theory. On the other hand, it states the contextual nature of the represented knowledge, by defining ontologies as *partial account* of reality. The *shared* aspect is not (explicitly) taken into consideration.

A wide range of definitions concentrate on the *content* of ontologies from an *engineering perspective*. Emerged from the knowledge base community, these definitions consider ontologies as means to specify a hierarchically structured vocabulary of a particular domain and the rules for *"combining the terms and relations to define extensions of the vocabulary"* [154]. Ontologies are then used as *schemas for building knowledge bases* in order to simplify the integration and reuse of these systems. The consequences of this engineering-oriented view upon ontologies are twofold: they implicitly give hints on the mandatory content of ontologies and on the ways to build them: ontologies consist of concepts/terms, relations, rules

etc., while concepts and relations might be organized in taxonomies. Accordingly, in order to build an ontology the knowledge engineer has to specify its vocabulary and proceed by identifying concepts, organizing them in a hierarchy and linking them by means of relationships and constraints. Furthermore, it is explicitly stated that ontologies are intended to be *reused* and *shared* among software applications, a fact which was not explicitly taken into account by the logics-centered approaches such as that by Guarino [86].

Given the multitude of application scenarios and the complementary points of view with respect to the exact meaning of ontologies we currently assist at a "mitigation" of the original definitions towards a more interdisciplinary, unified understanding of the term. Since ontology-like conceptual models have a long-standing tradition in computer linguistics (thesauri), database design (ER diagrams), software engineering (UML diagrams, object models) or eCommerce (product taxonomies, business rules), a recent trend in identifying whether a certain conceptual structure is or is not an ontology is to enumerate the mandatory conditions for the former [225]. In order to distinguish ontologies from other (related) conceptual structures Uschold and Jasper introduced two criteria [225]:

- an ontology *defines a vocabulary* of terms.
- an ontology *constrains the meaning of the domain terms* by indicating how ontology concepts are defined and are inter-related to a specific domain structure.

2.1.2 Types of ontologies

A further attempt to clarify the partially divergent views upon ontologies was to classify them by various dimensions:

1. **Formality**: Uschold and Grüninger distinguish among four levels of formality [223]:
   - highly informal: the domain of interested is modelled in a loose form in natural language.
   - semi-informal: the meaning of the modelled entities is less ambiguous by the usage of a restricted language.
   - semi-formal: the ontology is implemented in a formal language.
   - rigorously formal: the meaning of the representation language is defined in detail, with theorems and proofs for soundness or completeness.

   Most of the currently available (Web) ontologies can be included to the second and third category. Wand and Weber restrict to three levels of formality [229]:

   - informal
   - semi-formal, and
   - formal ontologies

   Again, most of the currently available sources usually associated to the word *ontology* can be ordered to the categoriy of *semi-formal* models. Lastly, McGuinness defines an *"ontological continuum"* specifying a total order between common types of models [138]. This basically divides ontologies (or ontology-like structures) in *informal* and *formal* as follows:

- **informal models** are ordered in ascending order of their formality degree as *controlled vocabularies*, *glossaries*, *thesauri* and *informal taxonomies*.
- **formal models** are ordered in the same manner: starting with *formal taxonomies*, which precisely define the meaning of the specialization/generalization relationship, more formal models are derived by incrementally adding *formal instances*, *properties/frames*, *value restrictions*, *general logical constraints*, *disjointness*, *formal meronimy* etc.

In the first category we usually encounter *thesauri* such as WordNet [147], *taxonomies* such as the Open Directory[2] and the ACM classification[3] or various eCommerce standards [60]. Most of the available Semantic Web ontologies can be localized at the lower end of the formal continuum (i.e. as *formal taxonomies*), a category which overlaps with the semi-formal level in the previous categorizations. However, the usage of Semantic Web representation languages does not guarantee a certain degree of formality: while an increasing number of applications are currently deciding to formalize domain or application-specific knowledge using languages such as RDFS or OWL, the resulting ontologies do not necessarily commit to the formal semantics of these languages.[4] By contrast, Cyc [186] or DOLCE [158] are definitively representative for the so-called *heavyweight ontologies* category, which corresponds to the upper end of the continuum.

2. **Shareability**: due to the difficulties encountered in achieving a consensual conceptualization of a domain of interest, most of the ontologies available today reflect the view of a restricted group of people or of single organizations with this regard. Standard classifications such as the Open Directory,[5] classifications of job descriptors, products, services or industry sectors have been developed by renowned organizations in the corresponding fields. Due to this fact, these ontologies are being expected to be shared across a wide range of applications. However, many Semantic Web ontologies have been developed in isolated settings without an *explicit* focus on being shared across communities or software platforms. Given this state of the art we distinguish among four levels of (expected) shareability:
   - **Personal ontologies**: they are the result of an individual development effort, reflecting the view of the ontology author(s) upon the modelled domain. Personal Semantic Web ontologies are published online and might be accessed by interested parties, but their impact is limited, as there is no explicit support for them being reused in other application contexts. Depending on the complexity of the ontology, they still might achieve a broad acceptance among a large user community. The best example for this behavior is the FOAF ontology.[6] The simple ontology describing common inter-human relationships, originally designed as a toy example, is meanwhile one of the most popular Semantic Web ontologies.

[2] `http://www.dmoz.org` last visited in May, 2006
[3] `http://www.acm.org/class/1998/` last visited in May, 2006
[4] Approaches such as OntoClean [89] tackle this problem by proposing general measures to discover inconsistencies in the usage of these formal semantics.
[5] `http://www.dmoz.org` last visited in May, 2006
[6] `http://xmlns.com/foaf/0.1/` last visited in May, 2006

- **Application ontologies**: they are developed in the context of a specific project for pre-defined purposes and are assumed to reflect the view of the project team (including the community of users) upon the modelled domain. Again, though Semantic Web technologies enable these ontologies to be made public on the Web, they are de facto intended to be used *within the original, project-related user community*. Their acceptance beyond these boundaries depends on the impact of the authoring authority in the specific area, but also on the general reusability of the ontologies. Many of the domain ontologies available so far can be included in this category.
- **Openly developed ontologies**: they are developed by a large, open community of users, which are free to contribute to the content of the ontology. The ontology, as a result of continuous refinements and extensions, emerges to a commonly agreed, widely accepted representation of the domain of interest. The evolution of the Open Directory classification is a good example for collaborative, Web-based ontology development: the core structure of the topic classification, originally proposed by Yahoo![7] and used in slightly modified form by various search engines, was extended by users, who also played an crucial role in the instantiation of the ontology with Web documents. Another prominent example is the Gene Ontology [74].
- **Standard ontologies**: they are developed for standardization purposes by key organizations in the field, usually being the result of an extended agreement process in order to satisfy a broad range of requirements arisen from various user communities. The majority of standard ontologies currently available are situated in the area of eCommerce: The United Nations Standard Products and Services Codes Ontology UNSPSC,[8] the RosettaNet classification[9] or the North American Industry Classification System NAICS.[10]

3. **Scope**: according to [86] ontologies can be classified into four categories:
   - **Upper-level/top-level ontologies**: they describe general-purpose concepts and their properties. Examples of upper-level ontologies are the Top-Elements Classification by Sowa [202], the Suggested Upper Level Merged Ontology SUMO [177] or the Descriptive Ontology for Linguistic and Cognitive Engineering DOLCE [158].
   - **Domain ontologies**: they are used to model specific domains such as medicine or academia. A typical example in this area is the Gene Ontology [74].
   - **Task ontologies**: they describe general or domain-specific activities.
   - **Application ontologies**: they are instantiations of domain ontologies having regard to particular application-related task ontologies and application requirements.

   Other authors mention an intermediary category called **core ontologies**, which cover the most important concepts in a given domain. For example, the Semantic Network

[7] `http://dir.yahoo.com/` last visited in May, 2006
[8] `http://www.unspsc.org` last visited in May, 2006
[9] `http://www.rosettanet.org` last visited in May, 2006
[10] `http://www.census.gov/epcd/www/naics.html` last visited in May, 2006

in UMLS contains general medical concepts such as disease, finding, syndrome, thus being a core medical ontology. Others differentiate between **application domain** and **application task ontologies** [111]. The former instantiates general-purpose domain knowledge to particular application constraints, while the latter corresponds, similar to the **application ontologies** introduced by Guarino [86], to a combination of domain-relevant declarative and procedural knowledge.

A last category of ontologies, which was not covered by the classifications mentioned so far, are the so-called **meta-ontologies** or (knowledge) **representation ontologies**. They describe the primitives which are used to formalize knowledge in conformity with a specific representation paradigm. Well-known in this categories are the Frame Ontology [83] or the representation ontologies of the W3C Semantic Web languages RDFS and OWL.[11].

4. **Reusability**: besides sharing, ontologies were originally intended as means to efficiently *reuse* different types of declarative knowledge. As in other areas of computer science, there is a trade-off between reusability and usability, since reusability usually implies efforts to adapt a general-purpose artifact to particular settings. Klinker and colleagues [111] further refine the ontology classification by Guarino [86], ordering ontology scopes with respect to their reusability potential, as illustrated in Figure 2.1.

5. **Representation paradigm**: ontologies differ in the representation language and paradigm, since a wide range of these sources emerged in a pre-Semantic Web era. In order to overcome this syntactic and semantic barrier a plethora of approaches investigate the compatibility between different formalisms [59], while the aforementioned *representation ontologies* are intended to capture these differences explicitly. The most popular representation paradigms regarding ontologies are Frames [83], Description Logics [176] and UML-MOF.[12] In the context of the Semantic Web ontologies are envisioned to follow the second approach. As a novel feature, they are represented in standardized, Web-suitable languages such as OWL [176] and RDFS [22] so that they can be shared and put into widespread use for humans and machines in a networked environment.

Due to these differences ontologies originated from and closely related to particular communities can not be optimally shared or reused in an operationalized manner in the open environment of the Web. This state of the art requires that potential ontology users are made aware of the (currently limited) reusability of these resources and that ontology engineering platforms are provided with techniques to deal with their heterogeneity with respect to the aforementioned dimensions (see Chapter 3 for a longer discussion on this topic).

We now turn to a description of recent methodologies and methods for building and managing ontologies.

[11] `http://www.w3.org/2000/01/rdf-schema`, `http://www.w3.org/2002/07/owl` last visited in May, 2006

[12] `http://www.omg.org/technology/documents/modeling_spec_catalog.htm` last visited in August, 2006

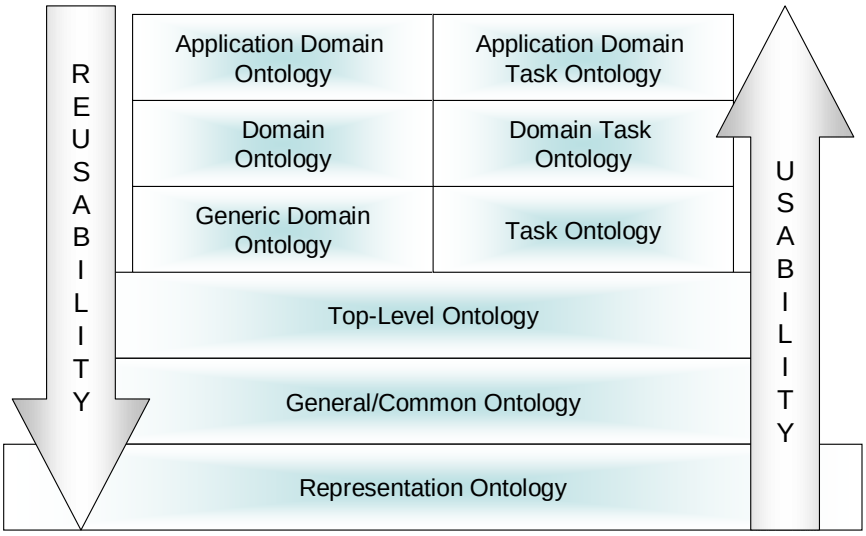


Figure 2.1: Classification of Ontologies as regards Their Reusability [111]

2.2 Ontology Engineering Methodologies and Methods

Ontologies, as a means for a shared understanding of knowledge and a way to represent real world domains, are a core technology for efficiently developing high-qualitative knowledge-based systems [60]. In particular they are facing a growing popularity with the emergence of the Semantic Web; besides providing standard Web-suitable representation languages such as RDF(S), OWL and SWRL, the Semantic Web community also encourages the development of methods and tools to build, maintain and reuse Semantic Web ontologies [44, 63, 80, 162] or algorithms to manage them in terms of matching, merging or integration [162, 196]. Due to the challenging and resource-intensive nature of the ontology building process special attention has also been paid to automatic methods to assist this process such as those based on the linguistics-based acquisition of domain-specific knowledge from document corpora [164]. The sum of these initiatives are referred to as *ontology engineering* [80].

The goal of this section is to present the current state of the art in ontology engineering, by giving an overview of some of the most outstanding methodologies and methods emerged in the last decades in this area. For a consistent classification of these research contributions we employ the terminology introduced by Gómez-Pérez and colleagues in [80]: this proposal, based on the IEEE standard for software development [104], differentiates between *management*, *development-oriented* and *support activities*. The former include typical control and quality assurance activities encountered across various engineering disciplines. Development-oriented activities strictly relate to the process of *ontology building*, while support activities aim at aiding the engineering team in this endeavor by means of documenting and evaluating specific stages of the development process or by extracting relevant ontological knowledge from external (semi-structured) resources.

2.2.1 Methodologies and Methods for Ontology Development

In terms of process models several engineering methodologies were elaborated in the last decades [64, 66, 84, 85, 105, 125, 179, 201, 209, 214, 226]. Apart from minor differences with respect to the level of detail and the underlying application environment, the majority of them propose the following development-oriented activities for ontology building (Figure 2.2):

1. **Domain/Requirements analysis**: analysis of the application domain with respect to a set of pre-defined requirements. In this phase one puts together a list of *competency questions*, describing the capabilities of the target ontology in the intended application scenario. This step also includes *knowledge acquisition* in terms of re-usage of existing ontological sources or performing ontology learning operations. If such techniques are being used to aid the engineering process, the resulting ontologies—acquired from texts or by reuse—are to be customized to the application setting in the conceptualization/implementation phases. The result of the domain analysis might be specified in form of an *ontology requirements specification document*, as introduced by the OTK-Methodology [214]. The methodology by Grüninger [85] takes the requirements specification phase a step further and suggests a formal representation of the emerging requirements—in their case (informal) competency questions—in terms of First Order Logic in order to allow an automatic evaluation of the final ontology in later phases.

2. **Conceptualization**: on the basis of the competency questions the engineering team models the application domain in terms of ontological primitives, e.g., concepts, relations, axioms.[13] The result of this step is a conceptual model, which, though being language-independent, already commits to a particular (formal) knowledge representation paradigm (e.g., Frames). In order to reduce the complexity of this process phase the engineering methodology by Fernandéz-López [62] distinguishes between *conceptualization* and *formalization*.

3. **Implementation**: the conceptual model resulted from the previous step is implemented in a (formal) representation language with adequate expressivity. The selection of a representation language for the implementation is also affected by the availability of support tools and by application constraints. Reused ontologies and those generated by ontology learning are potentially translated to the target representation language and integrated into the new context.

4. **Evaluation**: the preliminary ontology is evaluated against the application requirements. The evaluation may be performed automatically, if the competency questions are represented formally, or semi-automatically, using specific heuristics and human judgement. The result of the evaluation is reflected in a set of modifications/refinements at the requirements, conceptualization or implementation level.

5. **Population**: this step deals with the alignment of concrete application data to the implemented ontology. The alignment of the instance data to the final ontology is typ-

[13] Depending on methodology and representation language these ontological primitives might have different names, e.g., class or concept, relation or relationship, slot, axiom, constraint.

ically a non-trivial task; it might require complicated mappings between the original data schema and the conceptual model of the ontology, or procedures for extracting ontology-related information from unstructured data such as texts.

6. **Evolution and maintenance**: after the preliminary evaluation the ontology is integrated into the target application setting. Further modifications or even complex re-engineering tasks are performed in conformity with new user requirements, updates of the reused sources or changes in the modelled domain.

Depending on the ontology life cycle underlying the engineering methodology, the aforementioned steps are to be seen as a sequential workflow, as parallel activities or a combination of both. METHONTOLOGY [62], which applies prototypical engineering principles, considers *knowledge acquisition*, *evaluation* and *documentation* as being *support activities* complementary to the main development process. Other methodologies, usually following a waterfall model, consider the three as part of a sequential, but iterative engineering process (cf. Figure 2.2). In addition to the illustrated process steps, the OTK-Methodology [214] introduces an initial *feasibility study* in order to assess the risks associated with an ontology building attempt and to specify strategies to handle such critical situations.

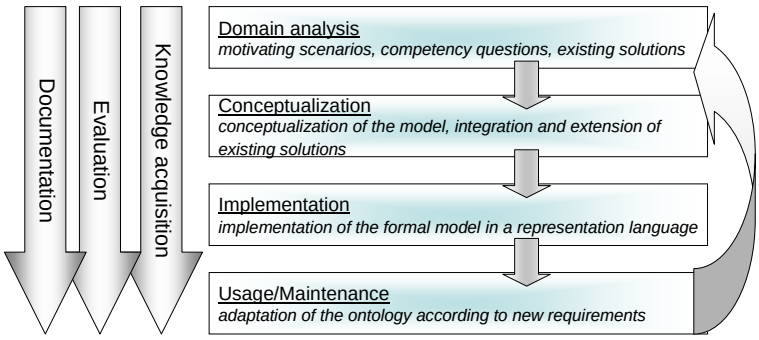


Figure 2.2: Ontology Engineering Process

The process model described above gives a very general overview on how ontology development is supposed to be performed in an application- and domain-independent manner. However, in order for the engineering methodology to represent a real added value to the ontology developers, it has to be further elaborated. Firstly, the whole range of dependencies among single process steps should be described in detail in order to maximize the flexibility of the process model and to allow ontology engineers to choose the development workflow which optimally fits their needs. Secondly, phases such as *knowledge acquisition* or *ontology evaluation* imply highly complex automatic methods and techniques, which have to be rigourously recorded in the process model [205]. Furthermore, the general-purpose model (as depicted in Figure 2.2) does not take into account issues such as *collaboration*, *distributed engineering*, or *scalability* which are fundamental characteristics of an ontology engineering process performed in the open, dynamic setting of the Web. Approaches such as [58, 181, 216] aim at coping with the former two issues and give guidelines on how to collaboratively built dynamic domain models dealing with volatile requirements. Building and using very large ontologies, like those typical for e.g., the medical domain, arise new

questions for the engineering process, since they are characterized by huge resource needs and long-standing life cycles. Methodologies such as [125, 179, 216] address this problem explicitly.

Ontology development is solely the first step an ontology life cycle. Once the ontology has been developed, it becomes subject of continuous refinements and evaluation procedures, which aim at updating it to new user-/domain-specific criteria and at applying it to new application scenarios, respectively. The evolution of an ontology during its usage in the designed application setting is explored in approaches such as [110, 208], which propose methods to deal with versioning and user-driven ontology maintenance. Methodologies and methods for using and reusing ontologies in new application contexts are discussed in the next section.

2.2.2 Methodologies and Methods for Ontology Use and Reuse

As mentioned at the beginning of this section the ontology engineering process consists of *management*, *development* and *support activities*. The latter include a series of activities performed *in parallel* to the core ontology development, which was introduced above. In this section we focus on activities intended to *manage* existing ontologies for reuse purposes, in form of *evaluating*, *matching*, *merging*, *aligning*, *mapping* or *integrating* them.

Ontology Reuse Methodologies

An extended account of the reuse process has been made by Pinto and Martins [180]. Differentiating between merging and integration, the authors give a general description of the phases and activities of the second one, where integration is defined as *"the process of building an ontology on one subject reusing one or more ontologies in different subjects"*. Reusing ontologies of the same domain is termed to *"merging"*, which is studied from a methodological point of view in [73]. In the following we will describe the process models underlying these two approaches.

The *integration process* [180] is composed of several phases and activities, which might take place sequentially or along the complete life cycle of ontology building:

1. **Identify integration possibility**: feasibility study with respect to the possibility of reusing ontological sources within the development framework.

2. **Identify modules**: decomposition of the domain into thematic clusters.

3. **Identify assumptions and ontological commitments**: specification of the ontological commitments of the building blocks identified in the previous step, ensuring compatibility.

4. **Identify knowledge to be represented in each module**: identification of a preliminary list of ontological primitives to be included in each module.

5. **Identify candidate ontologies**: this step is further divided into two sequential activities: *finding* the ontological sources, followed by the *choice upon the relevant* ones. The relevance assessment is performed using a checklist of strict and desirable requirements with the goal of eliminating the candidate ontologies who are definitely

not useful in the new setting. The remaining ontologies are subject of a more deeper evaluation along the process.

6. **Get candidate ontologies**: in order to preserve the generality with respect to representation languages, the approach proposes an integration of ontologies at knowledge level, while further material such as implementations and documentation may be used to complement the general-purpose, knowledge level representation. While most of the ontologies are available in some representation language, the conceptualization is obtained as a result of a re-engineering process. At this level, the process model also foresees the possibility of translating between representation languages, if differences between source and target implementation languages occur.

7. **Analyze candidate ontologies**: the analysis of the candidate ontologies, which are potentially relevant for the target domain, is the first step towards the final choice on which ontological sources will participate to the integration process. The analysis involves a *technical evaluation* by domain experts and an *user assessment* by ontology engineers. The former focuses on the content of the candidate ontologies and identifies necessary modifications, while in the latter the ontology engineers study the quality of the modeling decisions underlying the ontologies (e.g., well-balanced structure, homogeneous granularity and level of detail, coherent design rationales). The results of the analysis are used as decision support for the choice to be made in the next step.

8. **Choose source ontologies**: at this stage a decision has to be made among the candidate ontologies that passed the strict requirements and among those that were best ranked during the analysis performed by domain experts and engineers. Due to the complexity of this multi-criteria choice, this step is further sub-divided into a *selection* of the most appropriate remaining candidates and the *verification* of their compatibility and completeness. The first sub-stage is a continuation and a summarization of the analysis described above. By means of a set of general, development and content-oriented features, the engineering team chooses the source ontologies to be used. However, this decision is conditioned by the compatibility check, in conformity with which appropriate candidates are not integrated into the target ontology if they produce syntactic or semantics incompatibilities.

9. **Apply integration operations**: [180] defines a set of 39 basic integration operations and additional 12 composed ones in order to specify the aggregation of the knowledge content of the ontologies to be reused. The result of the integration may be subject to further modifications according to pre-defined design criteria such as modularization, minimization of the semantic distance within the same level of an hierarchy or naming conventions.

10. **Analysis of the resulting ontology**: the target ontology is finally evaluated against general-purpose design [84] or evaluation criteria [79].

A comprehensive study on reusing ontologies sharing the same domain is provided in the ONIONS (Ontologic Integration of Naïve Sources) project [73]. The result of the project is a detailed methodology on merging existing ontologies for reuse, which was applied in

the medical domain. The methodology, founded on philosophical aspects of the theory of meaning, is divided into six phases:

1. **Corpus creation**: in this step the focus is on finding relevant domain sources (e.g., classifications, ontologies, thesauri) in order to collect a valid terminological corpus for the target ontology.
2. **Taxonomic analysis**: local specialization/generalization relationships between concepts from the terminological corpus are evaluated and refined.
3. **Local source analysis**: the taxonomical structures are further analyzed in order to identify potential modelling failures such as inconsistent classification criteria or heterogeneous levels of detail between different fragments of the ontological sources. Since most of the sources available are not represented at a sufficient level of formality, the meaning of their concepts is mostly expressed implicitly in names or free text definitions, and has to be manually encoded in a formal, explicit way in order to create formal ontologies. The result of this step is a list of taxonomies, with explicit subsumption definitions.
4. **Multi-local source analysis**: local free-text descriptions are merged towards multi-local conceptual definitions or linked to general pieces of knowledge.
5. **Ontology library building**: the results of the local and global source analysis are integrated into an ontology library containing these conceptual definitions.
6. **Ontology library implementation and integration**: re-classification and validation of the library.

Further reuse methodologies often emerged in relation to specific application settings, thus being less general and comprehensive that the aforementioned proposals. Besides, evidence of reusing existing ontologies is attested in a number of case studies, in which the underlying methodology is usually left in the background of the work (see Chapter 3 for a discussion of the case studies).

For the remaining of this section we will take a deeper look at methods aiming at supporting particular aspects of ontology management. These aspects are a constitutive part of ontology reuse.

Ontology Discovery Methods

Finding ontologies on the Web has been marginally addressed by current approaches to ontology engineering. However, while there is no method describing how to spot online available ontological resources, ontology engineers might use one of the few ontology repositories available online as starting point for the completion of this task:

Protégé Library: collection of OWL ontologies hosted at the Protégé development group at the Stanford University. The library contains several dozens of ontologies freely submitted by Protégé users. The resources are accompanied by a short textual description.[14]

[14] `http://protege.stanford.edu/plugins/owl/owl-library/` last visited in May, 2006

Ontolingua Server: library of Ontolingua and KIF ontologies created in the early nineties at the KSL Group at the Stanford University. While the repository seems to have attracted considerable attention a decade ago—being mentioned as a reliable and useful content source in several case studies on reusing ontologies—its contents could not be accessed as for February 2006.[15]

SchemaWeb: recent library of several hundreds of Semantic Web ontologies, currently still under development. The ontologies are classified by name, description, namespace, location, Web site and contact person.[16]

Cyc Library: the most comprehensive common-sense knowledge base currently available, organized as a collection of so-called *"microtheories"*. An OWL version is under development.[17]

DAML Ontology Library: the most popular ontology repository, originated in the context of the DAML working group. Ontological resources (282, in total, primarily represented in languages like DAML+OIL, RDFS and OWL) are classified by URI (location), submission date, keywords, Open Directory category, class and property labels, used namespaces, funding source and submitting organization. For each ontology the user is provided with a synopsis of its most important metrics: the number of classes, properties, axioms and instances.[18]

OntoSelect: ontology repository for continuously discovering and managing Semantic Web ontologies. The over 800 resources (as for February 2006) are classified by name, format, natural language, as well as labels of ontological primitives. Additionally the service computes statistical metrics on the registered ontologies: the number of classes, properties, labels and included ontologies.[19]

Onthology: a novel repository of Semantic Web ontologies, which are classified according to the emerging ontology metadata standard **OMV** (cf. Chapter 5). The repository, which is currently under development, includes services for semantically searching and browsing the submitted ontologies and for the user-driven rating of their quality.[20]

Swoogle: a Semantic Web search engine, ranging, as stated by its authors, over more than 10.000 Semantic Web ontologies. These are not further classified.[21]

This list points out two important aspects about ontologies in general and ontology reuse in particular: firstly, the ontology engineering community has recognized the importance of providing repositories of (potentially reusable) knowledge components in early stages of its existence; secondly, this interest is shared by the newly emerging Semantic Web community which started developing dedicated ontology repositories in the last three to five years. A discussion of the quality and maturity of these approaches is given in Chapter 3.

[15] `http://www.ksl.stanford.edu/software/ontolingua/` last visited in February, 2006
[16] `http://www.schemaweb.info` last visited in May, 2006
[17] `http://www.opencyc.org/` last visited in May, 2006
[18] `http://daml.org/ontologies` last visited in May, 2006
[19] `http://views.dfki.de/ontologies/` last visited in February, 2006
[20] `http://www.onthology.org/` last visited in May, 2006
[21] `http://swoogle.umbc.edu/` last visited in May, 2006

Ontology Evaluation Methods

Ontology evaluation is not a fully developed area of ontology engineering yet (cf. [114] for an up-to-date overview of the methods). Though several approaches propose partial answers to the evaluation question, there is no commonly agreed, complete or of practical relevance methodology which describes the evaluation process both from a *general-purpose* and a *usage-related* perspective. The majority of the evaluation approaches proposed in the literature introduce methods to evaluate ontology content, i.e. to estimate the quality of the conceptual model independently of the fitness of use of the ontology under given circumstances:

OntoClean: the methodology is based on philosophical notions for a formal evaluation of taxonomical structures [89]. Core to the methodology are four fundamental ontological notions: rigidity, unity, identity and dependence, which are used to automatically verify the correctness of a taxonomy. The approach does not take into account usability aspects of ontologies, thus founding limited applicability in real-world scenarios.

ODEval: operates on taxonomical structures, allowing the detection of potential inconsistencies and redundancies for RDF(S), DAML+OIL and OWL ontologies [79]. The authors propose a list of general-purpose quality criteria for the content of an ontology, which will be discussed in detail in Chapter 5.

OntoManager: focuses on the usage of an existing ontology in a concrete application scenario and pragmatically extrapolates statistic data obtained from monitoring ontology users in order to detect potential errors or limitations of the ontology.[22] Though it is easy to apply by the end users, the accuracy of the results depends on amount of the monitored data and its accuracy.

Orthogonal to general-purpose quality of ontological content, a core requirement for systematically reusing existing ontologies in *concrete application contexts* is the availability of methods to quantitatively assign their relevance in respect to a set of application requirements. Most of the ontology evaluation approaches proposed so far do not address this issue, thus allowing ontology engineers and users to get an idea about the general-purpose quality of a given ontology, which might not necessarily correspond to its perceived quality or to its fitness of use. Assessing the usability of an ontology in a target application context is addressed briefly in [180]. The authors identify a number of issues relevant on this matter, such as compatible domain, representation language etc. Another exponent in this second category is OntoMetric [128], a framework for selecting ontologies. Provided a set of candidate ontologies, OntoMetric computes a quantitative measure of every element of this list using a framework of 160 features grouped in several dimensions, which describe the ontologies to be evaluated. After specifying the objectives of the application the ontology engineers build a decision tree containing ontology characteristics required in the application setting. The suitability value of each candidate ontology is computed by comparing its features with the nodes of the decision tree, taking into consideration the relativity relationships between its criteria. In addition to these approaches, we find approaches aiming at evaluating the a-posteriori usage of an ontology for a specific task such as semantic annotation of texts [19, 146, 203].

[22] `http://ontoware.org/projects/ontomanager` last visited in May, 2006

Ontology Matching, Merging and Alignment Methods

Due to the divergent opinions concerning the exact meaning of the term "ontology" and to the lack of an established quality framework for ontology engineering, most of the ontologies available today can not be easily reused in new application contexts (see Chapter 3 for a detailed discussion of the subject). Under these circumstances existing ontologies require considerable customization in form of matching, merging, mapping or alignment (cf. Chapter 1 for a terminological clarification). Furthermore, the data integration problem, which is relevant for many years at the enterprize level, has become even more stringent with the evolution of the current Web and the upcoming Semantic Web. Ontologies explicitly describing data repositories across the network should be compared and integrated into a common structure in order to cope with the heterogeneity, abundance and distributed nature of the data in a feasible manner. The importance of these issues is reflected by the high number of algorithms, which have been proposed and applied in certain application settings in the last decades [51, 54, 76, 130, 139, 143, 156, 183, 210]. Comprehensive studies, surveys and classifications on this topic are given for example in [50, 76, 184].

Each of these methods share the common goal of finding commonalities among a set of ontologies and the affiliated instance data. Once similarities have been found, merging algorithms generate a unified target ontology, obtained by aggregating concepts from the source ontologies which have been assigned a feasible similarity ranking. By contrast, mapping algorithms restrict to explicitly defining the relationships between these concepts. Depending on the outcomes of the matching procedure the mappings between concepts presumed to refer to the same real-world object may have various cardinalities, from direct to n-to-m. These particularities influence to a considerable extent the merging methodology, which has to provide decision support for resolving potential ambiguities.

Usually one distinguishes between individual matching algorithms (e.g., FCA-MERGE [210] or S-Match [76])—applying only a single method of matching items e.g., linguistic or taxonomical matchers—and combinations of the former ones, which intend to overcome their limitations by proposing hybrid solutions. A hybrid approach (e.g., Cupid [130]) follows a black box paradigm, in which various individual matchers are melt together to a new algorithm, while the so-called composite matchers allow an increased user interaction (e.g., GLUE [54],COMA [51], CMC [219]). Given the open nature of the Web environment, in which the emergence of a unique ontology for a given application domain is considered both unrealistic and unnatural, application interoperability depends directly on consistent mappings between ontologies adopted by inter-communicating services. Approaches coping with this problem propose a (formal) specification of the semantic overlap between ontologies and integrate matching techniques to automatically discover mapping candidates [53, 56, 107, 131, 160].

2.3 Application Scenarios for Ontologies

Ontologies are confronted with an ever-growing popularity in numerous areas of computer science. The need for an enabling technology to formalize domain or application background information explicitly is emphasized by the huge amount of knowledge from various domains already available in terms of semi-formal (network-accessible or at least electronically avail-

able) ontologies [8, 10, 16, 49, 60, 64, 74, 78, 125, 133, 147, 175, 179, 185, 186, 201, 221, 227].

Typical usage scenarios for knowledge processing on the basis of ontologies can be found in numerous projects in the areas of information retrieval, knowledge management, eCommerce and recently the Semantic Web [60, 80, 113, 159, 163, 205]. In order to handle the explosively growing amount of information and the variety of forms in which this information might be available, ontologies improve existing systems in the aforementioned areas as they extend their functionality by replicating to a certain extent the human-specific *"understanding"* of the domain they are dealing with. Besides acting as a means to represent knowledge (i.e. as *explicit* specifications of conceptualization, as stated in the definition by Gruber [84]) ontologies can be viewed as mediators between applications (i.e. as *shared* conceptualization, in the previously cited definition). In this way they enable the interoperability of the applications integrating them—these systems communicate to each other using a commonly agreed vocabulary with an unambiguous meaning.

In this section we give a brief overview of four of the most important application fields of ontologies: information retrieval, knowledge management, eCommerce and the Semantic Web, describing the role of ontologies in each of them.

2.3.1 Information Retrieval

In an information retrieval (IR) application, ontologies are used to enable the usage of so-called *"semantic search"* methods so that the system may return more relevant or more precise results. Compared to common retrieval techniques, such as those based on statistical measures (word frequency analysis) or popularity rankings, an ontology-driven semantic search aims at improving the precision and recall of a retrieval system by retaining a machine-understandable representation of the concepts being searched for. On a technical level, the usage of the ontology is twofold: on one hand, it provides a classification structure to index the information items of a repository, be that the Web, or the corporate memory of an organization. On the other hand, it is used at run time in conjunction with reasoning services for query rewriting and ranking purposes.

Among the most representative applications in the area of ontology-based information retrieval we can enumerate:

OBSERVER: uses ontologies to integrate existing data repositories into a global information system for the Basque culture [144].

SHOE: applies ontologies to provide background knowledge that might help in disambiguating and interpreting the content of traditional Web sites [97].

OntoSeek: adopts a language of limited expressiveness for the description of structured information sources such as Yellow Pages or product catalogues and uses the Sensus ontology [216] to improve the precision and recall of content-based retrieval [88].

XSEarch: uses XML-formalized ontologies to refine queries expressed in common XML languages with a more richer representation of the application domain [40].

CORESE: is a Semantic Web search engine built upon conceptual graphs (CG) and RDF [43]. It enables the processing of RDF Schema and RDF statements within the CG formalism and has been applied in a variety of application settings ranging from knowledge management to eLearning and eHealth.[23]

2.3.2 Knowledge Management

Knowledge management solutions require ontologies in order to improve the accessibility of information items constituting the corporate memory of an organization, be that by a semantics-aware retrieval system or by integrating heterogeneous information systems [60, 159]. Further on, ontologies might be used as means to represent organizational knowledge [159, 204]. The structure of an enterprize, its business sector, as well as the network of suppliers, customers and partners, represented in ontological form, can be automatically integrated to adapt general-purpose systems to the particularity of the organization (cf. for instance the FRODO project [228]). Automatically generated knowledge portals are the most prominent example in this respect (as illustrated for example in the DECOR project [1]). Further prominent European initiatives with major industry involvement in this field are

On-To-Knowledge : aimed at developing methodologies, methods and tools which enable the development and deployment of ontologies within and across enterprizes. In this context ontologies are understood as a major pre-requisite for the alleviation of commonly known problems related to organizing and managing large heterogeneous information repositories [46, 214].

CoMMA : uses agents and ontologies to implement a corporate memory [72]. Similar to the previous projects ontologies are regarded not only as a technological means to realize KM applications but also methodologically as a system-internal representation of the corporate knowledge, which is collaboratively created and maintained by the employees.

2.3.3 eCommerce

Ontologies and ontology-based applications are a promising alternative to existing B2C and B2B solutions. While the role of ontologies in the B2C scenario is mainly related to meaningfully retrieving product information to interested customers, the B2B context needs ontology-based technologies primarily for mediation purposes. They promote a common information structure in form of pre-defined classifications of products, organizations, regions, services, etc. and an agreed vocabulary for all market participants—a terminology for representing products to be traded, and legal and financial constraints. Furthermore, as in other application areas, ontologies might be an instrument to build more sophisticated services such as product comparison, personalization etc. Among emerging eCommerce solutions realized on the basis of ontology we mention:

ALICE: a personal Web shopping assistant that integrates a variety of knowledge sources to customize the shopping experience to individual preferences [55].

[23] `http://www-sop.inria.fr/acacia/soft/corese/index.html` last visited in May, 2006

CHEMDEX: a Web-based market maker for the life science research industry, which uses an eCommerce ontology to create a uniform interface to product entries from various catalogues and to enable advanced product search and comparison.[24]

MULECO: The Multilingual Upper-Level Electronic Commerce Ontology designed by the CEN/ISSS Electronic Commerce Workshop describes high-level terms, which can be used as inter-connection platform for more specific business ontologies.[25]

We now turn to an overview of the most recent and significant application scenario of ontologies, the Semantic Web.

2.3.4 Semantic Web

The World Wide Web has caused a fundamental shift in the way we access information and services. However, the current Web is aimed mostly at people—information can be found in pages written in natural language and the functionality of many existing Web services is described in human-readable form. The drawbacks of this situation become evident when searching for precise information in a Web page—a case in which even the most sophisticated information extraction and page indexing techniques provided by current search engines still require exhaustive manual post-processing—or when complementary Web services have to be combined for joint usage—in this case standard technologies, including recent approaches in the areas of Web services discovery and composition, quickly reach their boundaries as well.

The Web of the next generation, introduced by Tim Berners Lee under the name *"Semantic Web"* [12], aims at dealing with such situations by augmenting its current information space with formalized knowledge and structured data which can be processed by computers. According to the Merriam Webster Online Dictionary[26] the word *"semantic"* is interpreted as *"of or relating to meaning in language"*. In case of the Semantic Web the term indicates that the meaning of the data on the Web can be made explicit or discovered not only by people, but also by computers. The semantics of the Web information content and the terms and requirements of Web service operation are represented formally and explicitly, are ubiquitously available and can be exchanged and extended both by humans and machines across the Web. In this way the Semantic Web enables computers to provide more help to people, being beneficial for companies and for private persons in the same time; for the former, since they can inter-operate with each other and supply integrated services, thus finding more customers, and for the latter, since they are provided a more personalized and powerful framework to get access to information and services.

In order for the Semantic Web to be realized, supporting standards and technologies are required for enabling machines to unambiguously *"understand"* the meaning of Web resources:

- **Languages** which are suitable to the open nature of the Web and allow a powerful and efficient knowledge representation,

[24] `http://www.chemdex.org/` last visited in May, 2006

[25] `http://www.cenorm.be/isss/Workshop/ec/MULECO/Documents/Muleco_Documents.htm` last visited in May, 2006

[26] `http://www.m-w.com/` last visited in May, 2006

- **Ontologies** to conceptualize shared views upon knowledge domains as well as means to manage them, and finally
- **Services** bringing the benefits of the Semantic Web to its users.

These aspects are illustrated in the Semantic Web architecture, which is depicted in Figure 2.3.

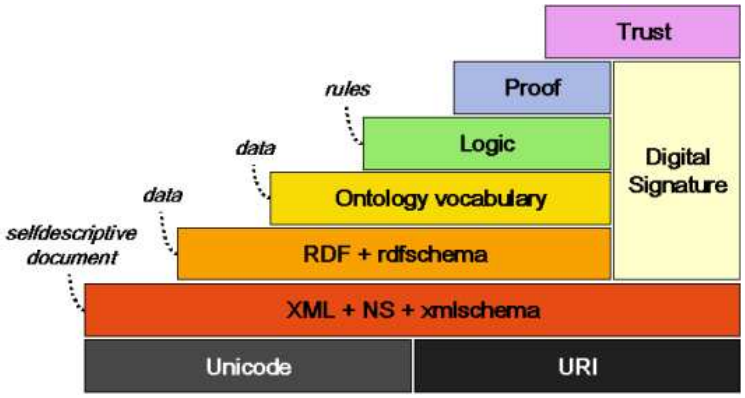


Figure 2.3: High-Level Architecture of the Semantic Web [11]

The first steps towards the realization of the Semantic Web have been made in the last decade through the standardization of a series of formal languages for the representation of Web knowledge: RDF(S) [95], OWL [176] and SWRL [100], and through the increasing dissemination of ontologies that provide a common basis for semantic annotation and support automatic inferencing for knowledge generation. Recent efforts in the area of Web Services propose methods and tools to represent Web Services in a Semantic Web-compatible manner in order to improve tasks like automatic service discovery or composition [61, 120, 152, 165, 200].

The Semantic Web is built on XML-based syntaxes which use URIs to uniquely identify Web resources. Resources can denote not only common Web documents, but any entity represented within a computer system (e.g., persons, physical objects, RDF statements—see the subsequent example) and are described by machine-processable metadata. Metadata is a collection of RDF statements of the type ⟨subject,predicate,object⟩, where the three fields can be individually referenced by means of URIs. This enables RDF to represent simple statements about resources as a graph of nodes and edges representing the resources, and their properties and values, respectively. We will illustrate the core RDF concepts and their meaning with a simple example.

We can annotate the present thesis (if available as a Web document) with information concerning its author, content etc. Examples of such statements could be:

1. **Elena Paslaru Bontas is the author of this thesis**
   The same statement in RDF:
   ⟨*myschema:PhDThesis http://purl.org/dc/elements/1.1/author http://mywebpage/Elena*⟩

The subject of this statement is this thesis, identified through the imaginary URI *myschema:PhDThesis*. The predicate *http://purl.org/dc/elements/1.1/author* is part of the Dublin Core metadata schema,[27] a vocabulary for authoring information, and denotes a standardized property, while the object *http://mywebpage/Elena*, the value of the property, represents the author Elena Paslaru Bontas.[28]

2. **This thesis describes the context ontology**
   In RDF we could relate these two Semantic Web resources, the thesis and the ontology it describes by means of the following statement:
   ⟨*myschema:PhDThesis http://purl.org/dc/elements/1.1/description http://nbi.inf.fu-berlin.de/research/swpatho/context/context.owl*⟩

   Again the predicate of the statement is part of the Dublin Core vocabulary, while the value of the object is the URL locating the context ontology.

3. **The email address of Elena Paslaru Bontas is "paslaru@inf.fu-berlin.de"**
   In RDF:
   ⟨*http://mywebpage/Elena myschema:mailto "paslaru@inf.fu-berlin.de"*⟩ The object in an RDF statement can be another RDF resource—represented by an URI or a literal, a simple XML-based datatype like in the present example—or a blank node (see below).

4. **The contact address of the author is "Takustr. 9, 14195 Berlin, Germany"**
   Structured information like addresses can be represented in RDF in aggregated form, by using so-called *blank nodes*, which denote anonymous resources, identified by automatically generated URIs:
   ⟨*http://mywebpage/Elena myschema:contact myschema:addressid1*⟩
   ⟨*myschema:addressid1 myschema:street "Takustr."*⟩
   ⟨*myschema:addressid1 myschema:number "9"*⟩
   ⟨*myschema:addressid1 myschema:zip "14195"*⟩
   ⟨*myschema:addressid1 myschema:city "Berlin"*⟩
   Here the URI *myschema:addressid1* identifies an anonymous resource which aggregates the address-related data.

5. **The author of the statement "This thesis describes the ontology context ontology" is Elena Paslaru Bontas**
   RDF applications sometimes need to say something about other RDF statements using RDF, for instance, to record information about when statements were made, who made them etc. In such cases RDF provides the *reification* mechanism, a means to obtain a reference to an RDF statement and annotate it with additional information:
   ⟨*myschema:statement1 rdf:type rdf:Statement*⟩
   ⟨*myschema:statement1 rdf:subject myschema:PhDThesis*⟩
   ⟨*myschema:statement1 rdf:predicate http://purl.org/dc/elements/1.1/description*⟩
   ⟨*myschema:statement1 rdf:object http://nbi.inf.fu-berlin.de/research /swpatho/context/context.owl*⟩

[27] `http://dublincore.org/` last visited in May, 2006
[28] We consider that the person in her role as a thesis author is identified by her web page.

⟨*myschema:statement1 http://purl.org/dc/elements/1.1/author http://mywebpage/Elena*⟩
Here we explicitly define an RDF statement as a resource in terms of its subject, predicate and object. Being a Web resource, the RDF statement is assigned an URI, which can be used to express further properties such as authorship.

To summarize the examples above, this RDF information can be represented in a graphical form as depicted in Figure 2.4.

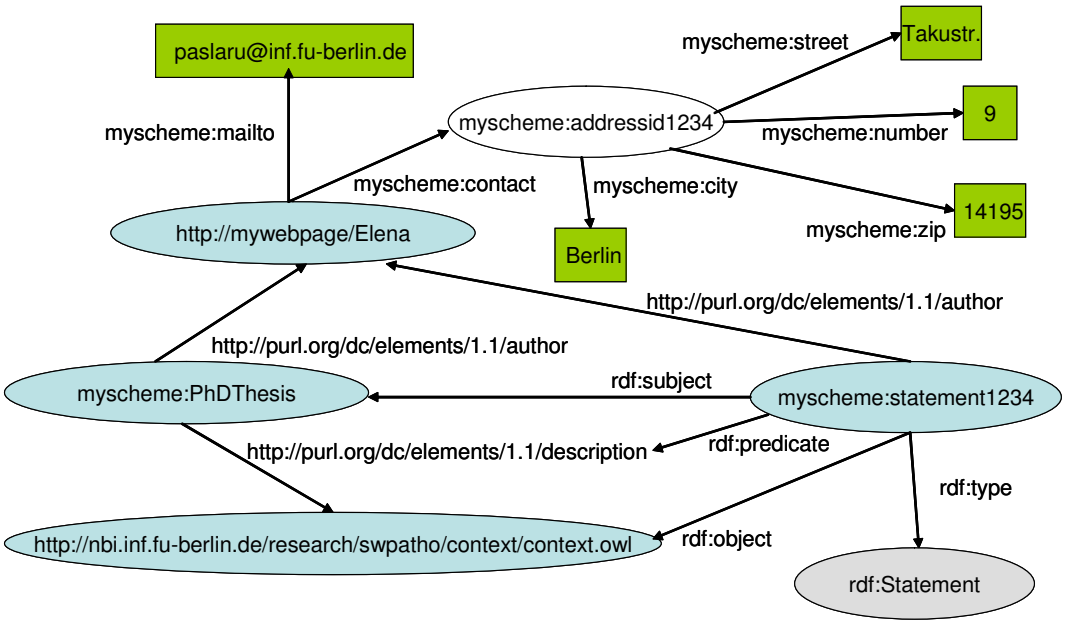


Figure 2.4: Graph Notation for RDF Annotations

RDF is considered to be the standard interchange format of the Semantic Web and is intended to be used as a simple, yet powerful annotation language for Web resources. The next layers in the Semantic Web language architecture add logically more expressive ontologies (e.g., OWL) and support for rules (for example SWRL [100]). RDF(S) and OWL are used to formalize common vocabularies for metadata—OWL allows even equivalence definitions between resources—thus increasing interoperability among applications. Besides, they re-define Web resources in terms of classes and properties with a well-formed semantics, which can be exploited by reasoners to generate implicitly formalized knowledge automatically. OWL is divided into three sub-languages, in order to maximize the trade-off between usability (with respect to expressivity) and computational properties (since more expressivity is achieved only at the expense of feasible computational properties). For simple knowledge representation tasks which involve classical subsumption, equivalence and restricted cardinality axioms ontology engineers are provided with the OWL Lite language. This language is expected to cover most of the expressivity needs in current Semantic Web applications, while offering efficient computational support. The OWL DL language extends the expres-

sivity of the former one, while still providing feasible properties as regards decidability. The OWL Full language is compatible to RDFS, thus being extremely powerful with respect to its expressivity, but it does not guarantee decidable computational features. OWL is complemented by SWRL, a rule-based knowledge representation formalism, which adds even more expressive power to OWL, but in the same time requires advanced reasoning capabilities to deal with the twofold (i.e. Description Logics and Rules) representation paradigm induced by OWL/SWRL knowledge bases.

We demonstrate the usage of ontologies and ontology representation languages by extending the previous example with additional information, which align the local vocabulary to external ones, and provide a more precise specification of the underlying domain, thus enabling more powerful information retrieval in a fictive publication repository.

1. **A PhDThesis is a special kind of Publication**
   Specialization and generalization hierarchies can be built in RDFS:
   ⟨*myschema:PhDThesis rdfs:subClassOf myschema:Publication*⟩
   In this way a query looking for publications of a given author would match not only to the documents which are explicitly defined (through annotations) as publications, but also to specific publication types such as PhD thesis or conference article.

2. **The concept "Publication" in the local schema is equivalent to the concept "Text" in the Dublin Core Vocabulary** Equivalence relationships between entities can be expressed in every of the three dialects of the OWL language:
   ⟨*myschema:PhDThesis owl:sameClassAs dc:Text*⟩
   Aligning the personal ontology to a standard model such as Dublin Core is the first step towards an increased syntactic and semantic interoperability. Ensuring syntactic compatibility enables, just as in the previous example, more flexible retrieval services, which are then able to go beyond a simple string matching-based functionality. The semantic interoperability allows applications handling the local ontology to better understand the meaning of the corresponding concept (in this case the PhDThesis), since the equivalence between concepts implies also an inheritance relationship by means of which the local concept is enriched with the properties externally defined for its equivalent.

3. **Every PhDThesis has a unique Author**
   By means of OWL we can refine the meaning of the entities in an ontology, this improving the knowledge background of the agents handling the modelled domain. For example, one can extend the simple example above with domain-specific knowledge such as the uniqueness of authoring. This enables agents to for example differentiate between PhDThesis and other types of publications, which might be multi-authored. This statement is formalized in OWL by defining the authoring property, which was introduced above, as being functional:
   ⟨*http://purl.org/dc/elements/1.1/author rdf:type owl:FunctionalProperty*⟩
   Note that this formalization does not imply that every PhDThesis is supposed to have at least one author. This information can be formalized using the OWL restriction

owl:minCardinality:
⟨*myschema:PhDThesis rdfs:subClassOf myschema:_x*⟩
⟨*_x rdf:type owl:Restriction*⟩
⟨*_x owl:onProperty http://purl.org/dc/elements/1.1/author*⟩
⟨*_x owl:minCardinality "1"* ⟩
In particular the class PhDThesis is declared to be a sub-class of an anonymous restriction class, which encompasses the desired additional information (i.e. the existence of an author).

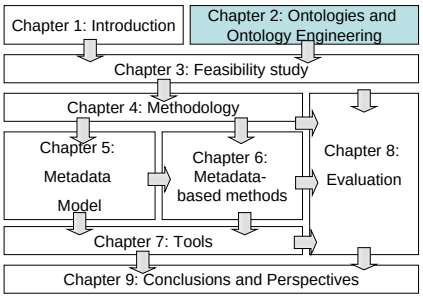
4. **A PhDThesis consists of several Chapters** The domain model is further refined in order to define typical (physical or conceptual) parts of a thesis. As in the previous example, extending the model supplies machines with deeper background knowledge on the application domain, thus enabling advanced, domain-tailored behavior.
   From a modeling point of view we introduce a new property *part_of* which relates entities to their parts:
   ⟨*myschema:PhDThesis myschema:part_of myschema:Chapter*⟩
   Modeling general-purpose relationships such as mereonimic ones is a topic with a long tradition in knowledge representation. For exemplification purposes we restrict ourselves to introducing a "symbolic" part-of relationship, which might be further specified or aligned to a standard model that rigorously defines such upper-level categories. The information that a thesis has at least one chapter can be formalized using a cardinality restriction as previously showed.

5. **Every chapter is related to at least one Topic**
   We associate topics (e.g., from a commonly-used topic category such as ACM) to thesis chapters in order to enable content-based search in the publication repository:
   ⟨*myschema:Chapter:hasTopic acm:Topic*⟩

6. **If a topic is relevant to a thesis chapter, then it is relevant for the thesis**
   The topic relevance is inherited along the part-of relationship from chapters to the complete thesis. OWL, as many other idioms used in knowledge representation, is restricted per default to is-a property propagation and subsumption reasoning, while additional part-of-focused inference support is expected to be available on the client side. In order to provide standardized, Web-compatible support for modelling this kind of information, recent research initiatives propose the rule language SWRL [100], which extends OWL with means to express deduction and dependency rules.

Despite their importance, the remaining layers of the Semantic Web are still at a much more immature stage. Issues of proof and trust are vital to the success of the Semantic Web, since one needs techniques to verify the quality and trustworthiness of the created metadata. The proof layer (Figure 2.3) is intended to embed languages and tools to prove whether statements created by arbitrary authors are true or not. The trust layer addresses the same issue from a different perspective—it should define mechanisms which, in correlation with digital signatures, enable the definition of provenance and reputation information for the resource metadata.

2.4 Summary



In this chapter we introduced the theoretical foundations of our research. After an incursion in the field of ontologies, we got closer to approaches describing how these conceptual structures can be built, managed, customized and reused. The last part of the chapter gave an overview of the four application sectors, which are expected to take benefit from using ontologies. In this context, we sketched the basic principles of the Semantic Web, which promotes technologies for representing and using ontologies to create a Web of machine-understandable information, and to mediate between services across this network.