3
Enterprise Knowledge Structures

Basil Ell, Elena Simperl, Stephan Wölger, Benedikt Kämpgen, Simon Hangl, Denny Vrandečić, and Katharina Siorpaes

3.1 Introduction

One of the major aims of knowledge management has always been to facilitate the sharing and reuse of knowledge. Over the years a long list of technologies and tools pursuing this aim have been proposed, using different types of conceptual structures to capture the knowledge that individuals and groups communicate and exchange. This chapter is concerned with these knowledge structures and their development, maintenance and use within corporate environments. Enterprise knowledge management as we know it today often follows a predominantly community-driven approach to meet its organizational and technical challenges. It builds upon the power of mass collaboration and social software combined with intelligent machine-driven information management technology delivered though formal semantics. The knowledge structures underlying contemporary enterprise knowledge management platforms are diverse, from database tables deployed company-wide to files in proprietary formats used by scripts, from loosely defined folksonomies describing content through tags to highly formalized ontologies through which new enterprise knowledge can be automatically derived. Leveraging such structures requires a knowledge management environment which not only exposes them in an integrated fashion, but also allows knowledge workers to adjust and customize them according to their specific needs. We discuss how the Semantic MediaWiki provides such an environment - not only as an easy-to-use, highly versatile communication and collaboration medium, but also as an

B. Ell (✉) • E. Simperl • B. Kämpgen • D. Vrandečić
Karlsruhe Institute of Technology, KIT-Campus Süd, D-76128 Karlsruhe, Germany
e-mail: basil.ell@kit.edu; elena.simperl@kit.edu; benedikt.kaempgen@kit.edu; denny.vrandecic@kit.edu

S. Wölger • S. Hangl • K. Siorpaes
STI Innsbruck, University of Innsbruck, Technikerstraße 21a, 6020 Innsbruck, Austria
e-mail: stephan.woelger@sti2.at; simon.scerri@deri.org; katharina.siorpaes@gmail.com

P. Warren et al. (eds.), *Context and Semantics for Knowledge Management*,
DOI 10.1007/978-3-642-19510-5_3,

integration and knowledge engineering tool targeting the full range of enterprise knowledge structures currently used.

This chapter is split into two parts. In the first part we undertake a comparative analysis of the different types of knowledge structures used by knowledge workers and enterprise IT systems for knowledge sharing and reuse purposes. In the second part we devise a comprehensive approach to develop, manage and use such structures in a collaborative manner. We present an ontology editor bringing together Web 2.0-inspired paradigms and functionality such as Flickr (http://www.flickr.com) and wikis to support laymen in organizing their knowledge as lightweight ontologies. Integration with related knowledge resources is exemplified through a series of methods by which arbitrary folksonomies, but also highly popular knowledge bases such as Wikipedia (http://www.wikipedia.org/) and Freebase (http://www.freebase.com/) are made accessible in ontological form. To further optimize the usability of knowledge structures – an issue which becomes particularly important in a non-expert-driven knowledge engineering scenario integrating various resources – we design techniques to check for common fallacies and modeling errors, which offer a solid baseline for cleansing the underlying knowledge base. The implementation is based on Semantic MediaWiki (http://semantic-mediawiki.org/), and has been deployed in the three case studies of the ACTIVE project which are introduced in Chaps. 9–11.

3.2 Enterprise Knowledge Structures and How Are They Used

The question of how to optimally capture and leverage enterprise knowledge has engaged the knowledge management community since its inception. As already discussed in the introductory section of this chapter, the prominence of this topic is reflected in the different types of conceptual structures which we can find behind the scenes of enterprise knowledge management platforms, a diversity which is multiplied by the wide spectrum of methodologies, methods and techniques proposed for their development, maintenance and use. In the present day, enterprise knowledge management essentially follows a community-driven approach, implementing solutions for crowdsourcing and social networking in order to optimize communication and collaboration – within the company and its ecosystem of business partners and end-customers – and knowledge sharing and reuse. In addition, formal semantics provide intelligent information management technology for capturing, accessing, managing and integrating knowledge. The approach is based on ontologies, knowledge structures whose (community-agreed) meaning is expected to be exploitable by machines, in particular via reasoning facilities by which implicit knowledge is derived and inconsistencies are detected.

In the following we illustrate how enterprise knowledge structures can be used, and the various trade-offs which are associated with the different types of structures, in terms of three motivating scenarios taken from the case studies.

3.2.1 Knowledge Management at an International Consulting Company

The first scenario is set in a large, knowledge-intensive enterprise – a consulting company – where employees collaborate around the globe on various topics to provide services to clients with best efficiency.

Most enterprise knowledge management systems are set up for the 'prototypical' user with no specific task in mind. Especially in a large enterprise context, employees need information for various different tasks. For instance, they may want to find information on previous projects, get an overview of a specific technology, or they may be interested in learning about a particular group within the company. These tasks are particularly relevant in the context of proposal development, by which a company creates a description of the products and services it is offering at an estimated cost to a potential customer.

Proposal writing follows standardized processes and procedures – giving instructions about the tasks to be undertaken, the information to be gathered, the documents to be created, etc. Equally important are less formalized practices – calling contacts that may have information on similar projects, or searching for similar proposals in the intranet. Often, information about previous projects cannot simply be obtained from a central data repository. This is due to the fact that many documents created within the context of a client project are client-proprietary, and may not be shared within the entire company. There are also many technical challenges related to the decentralized and heterogeneus nature of the enterprise IT landscape, and to the limitations of keyword-based information management technologies. Especially in an enterprise scenario, and in the context of a specific task, it will often be useful to retrieve actual facts rather than the documents that mention them. Such facts refer to entities, for example, to experts, locations, clients, other companies, and relationships among them. Which facts should be retrieved naturally depends on the task at hand: for instance, in the case of proposal development, one might want to find clients for which the company has submitted similar bids.

Enterprise knowledge structures are the backbone of such sophisticated information retrieval facilities. They capture enterprise domain knowledge at various levels of expressivity and formality. When choosing the most appropriate among these levels, it is important to weigh the advantages and disadvantages of heavyweight ontology-based approaches, supporting reasoning and full-fledged semantic search, vs. the additional costs associated with the maintenance and usage of the knowledge structure, which should be integrated into the daily workflow and allow user participation at large. Enterprise document repositories support bookmarking and tagging as means to describe the content of documents. The resulting conceptual structures contain knowledge which could prove extremely useful to create rich, formal ontologies to implement more purposeful information retrieval solutions.

3.2.2 Knowledge Sharing at a Large Telecom Operator

A similar scenario has been identified at a large telecom operator.

Operating in multiple projects is a reality of modern businesses. As part of their daily work knowledge workers interact with various systems, information sources and people. Their work is highly dependent on contextual dimensions as diverse as the customer, the status of the sales opportunity, current project issues, and the suppliers involved. To improve productivity, frequently used information such as contact and customer data and product documentation should be easily available; the knowledge worker should not have to search around for these things as they change from one working context to another. Furthermore, as the user resumes an earlier task, her working context should be restored without problem to the state it was before.

There is an abundance of information held within the company's repositories, much of which may not be easily accessible to technical consultants, solution consultants, and sales specialists. In addition there is a wealth of tacit knowledge which may not be being captured to best effect. The key problem here is that knowledge workers may not be aware of earlier solutions; it is possible that comparable solutions to similar problems are being worked on in isolation rather than in co-operation, or even that a particular problem has already been solved. A better awareness of the solutions to specific business problems and the business domains in which those solutions were applied should enable common patterns of solutions to be identified.

To support agile knowledge working, several knowledge management features are required: information such as contacts, relevant (technical) documentation, emails, and customer-specific information must be captured and easy retrieval must be enabled. Moreover, the context of a knowledge worker has to be captured. This involves modeling of general enterprise knowledge as well as appropriate knowledge representation formalisms, suitable from an information-management point of view, but also tangible for knowledge workers.

3.2.3 Process Optimization at a Digital Chip Design Company

The order of the design activities during chip design is hard to determine before process start. Usually a designer or a team decides on the best possible continuation of the activity flow in an ad-hoc manner during the process. Problems can occur in the case of goal changes, requirement changes, environment changes, etc.

It is important to collect data about the actual execution and sequences of design activities in several concurrent design project flows. Ideally, this should be supported by a knowledge management application that assists in eliciting the knowledge about how a sequence of design and verification activities is related to a particular type of a designed artifact, the configurations of used design tools, and the capabilities of design teams.

The company uses a modeling framework and an upper-level ontology for representing dynamic engineering design processes and design systems as process environments. The modeling approach is based on the understanding that an engineering design process can be conceived as a process of knowledge transformation which passes through several states. Each state is the state of affairs in which a particular representation of a design artifact or several representations are added after being elaborated by a design activity leading to this state. Evidently, the overall goal of a design process is to reach the (target) state of affairs in which all the representations are elaborated with enough quality for meeting the requirements. The continuation of the process is decided by choosing an activity from the set of admissible alternatives for that state. Engineering design processes are situated in and factually executed by the design system comprising designers, resources, tools, and normative regulations.

The ontology used in the chip design company is a core component of all processes. The ontology constantly evolves and its evolution needs to be supported by collaborative ontology engineering tools. The objective is to ensure that the enterprise knowledge structures and the proprietary ontology suite are aligned.

3.2.4 Trade-off Analysis

Enterprise knowledge structures vary with respect to a number of aspects, ranging from expressivity to size, granularity and modeling paradigm followed. These aspects influence not only the utility of (a category of) knowledge structures in a particular scenario, but has also direct consequences on the ways in which a knowledge structure is developed, maintained and used. This section aims to conduct a baseline analysis of the trade-offs implied by these aspects and to introduce methods which can be used to perform such an analysis in a systematic manner.

Particular attention is paid to the use cases discussed above. Considering the scenario within the large consulting company, enterprise knowledge structures can be used to allow for the implementation of intelligent knowledge organization and retrieval techniques. Questions related to the most adequate type of knowledge structure, its tangible benefits, and the associated development and maintenance costs are crucial to demonstrate the added value of the technology in this scenario. The maintainability of knowledge structures is, besides reuse, an essential aspect of the second scenario we investigate in the project. Here, the additional problem to be looked into is the extent to which reusability of existing knowledge structures is economically feasible. The chip design scenario leverages ontologies as means to capture domain knowledge and enable communication between designers. Cost-benefit-motivated quantitative and qualitative means are expected to optimize the ongoing ontology engineering process (see Chap. 4).

Trade-offs are specified along a number of dimensions used in the literature to classify and describe knowledge structures:

1. **Formality**: (Uschold and Grueninger 1996) distinguish among four levels of formality:

- Highly informal: the domain of interest is modeled in a loose form in natural language.
- Semi-informal: the meaning of the modeled entities is less ambiguous by the usage of a restricted language.
- Semi-formal: the knowledge structure 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.

(McGuiness 2003) defines a '*semantic spectrum*' specifying a total order between common types of models. This basically divides ontologies or ontology-like structures in *informal* and *formal* as follows (Fig. 3.1):

- **Informal models** are ordered in ascending order of their formality degree as *controlled vocabularies*, *glossaries*, *thesauri* and *informal taxonomies*. In this category we can also include *folksonomies*, sets of terms which are the result of collaborative tagging processes.
- **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*, *disjointness*, *formal meronymy*, *general logical constraints* etc.

In the first category we usually encounter *thesauri* such as WordNet (Fellbaum 1998), *taxonomies* such as the Open Directory (http://www.dmoz.org) and the ACM classification (http://www.acm.org/class/1998/) or various eCommerce standards (Fensel 2001). 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

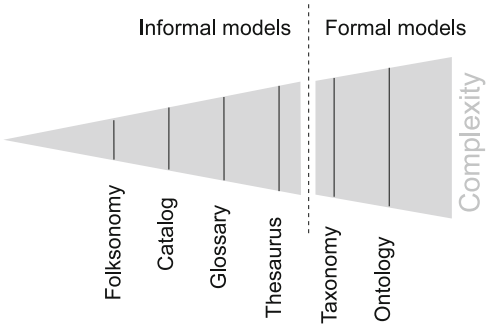

Fig. 3.1 Semantic spectrum (based on McGuiness [2003])

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. By contrast, Cyc (Lenat 1995) or DOLCE (Gangemi et al. 2002) are definitively representative for the so-called *heavyweight ontologies* category, which corresponds to the upper end of the continuum.

In (Vrandečić 2009b) we offer a complete formalization of all the above types of knowledge structures, and thus also how OWL2 (Grau et al. 2008) can be used to represent each of the other types besides ontologies. This allows us to classify knowledge structures automatically, and to check if they indeed meet the criteria of a specific type of knowledge structure. What is important here is that we can use any of the given structures without restrictions and nevertheless guarantee the integration of all these knowledge structures.

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. Standard classifications such as the Open Directory (http://www.dmoz.org), classifications of job descriptors, products, services or industry sectors have been developed by renowned organizations in the corresponding fields. Due to this fact, these knowledge structures are being expected to be shared across a wide range of applications. However, many of them 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**: the result of an individual development effort, reflecting the view of the author(s) upon the modeled 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.
- **Application ontologies**: developed in the context of a specific project for predefined purposes and are assumed to reflect the view of the project team (including the community of users) upon the modeled domain. Whilst under circumstances 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**: developed by a large, open community of users, who 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! (http://dir.yahoo.com/) 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 (Gene Ontology Consortium 2000).
- **Standard ontologies**: 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 ontology-like structures currently available are situated in the area of eCommerce: The United Nations Standard Products and Services Codes UNSPSC (http://www.unspsc.org), the RosettaNet classification (http://www.rosettanet.org) or the North American Industry Classification System NAICS (http://www.census.gov/epcd/www/naics.html). Another example is the FOAF ontology (http://xmlns.com/foaf/0.1/). The simple ontology describing common inter-human relationships enjoys significant visibility, not only as a result of the standardization efforts of the FOAF development team.

3. **Domain and scope**: according to (Guarino 1998) ontologies can be classified into four categories:

- **Upper-level/top-level ontologies**: they describe general-purpose concepts and their properties. Examples are the Top-Elements Classification by Sowa (Sowa 1995), the Suggested Upper Level Merged Ontology SUMO (Pease et al. 2002) or the Descriptive Ontology for Linguistic and Cognitive Engineering DOLCE (Gangemi et al. 2002).
- **Domain ontologies**: they are used to model specific domains in medicine or academia. A typical example in this area is the Gene Ontology developed by the Gene Ontology Consortium (2000).
- **Task ontologies**: they describe general or domain-specific activities.
- **Application ontologies**: they are extensions of domain ontologies having regard to particular application-related task ontologies and application requirements.

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 category are the Frame Ontology (Gruber 1993) or the representation ontologies of the W3C Semantic Web languages RDFS and OWL (http://www.w3.org/2000/01/rdf-schema, http://www.w3.org/2002/07/owl).

When describing the scope of an ontology, the types of knowledge that should be available to the engineering team to build the domain conceptualization are highly relevant. In principle, one can distinguish between ontologies capturing common and expert knowledge, and based on this distinction determine the composition of the team engineering a particular ontology.

4. **Representation language**: a wide range of knowledge structures 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, while the aforementioned *representation ontologies* are intended to capture

these differences explicitly. The most popular representation paradigms regarding ontologies are Frames, Description Logics and UML-MOF.[1]

On the Semantic Web, the classic trade-offs regarding expressivity have been decidability and complexity. For a language to be decidable, a reasoner can be implemented such that all questions that can be asked against a knowledge base that are expressed using that language have an answer. Decidability as a property of languages is highly desirable: it guarantees that all questions that can be asked can be answered, and that the associated reasoning algorithms are effectively implementable. Research in Description Logics explores the borders of decidability.

Besides decidability, which guarantees the effective implementation of reasoning algorithms, we further need to regard the complexity of the algorithms that can answer questions against the knowledge structures. In general it can be said that the more expressive a language is, the higher the complexity. Since neither expressivity nor complexity are continuous spectra, it can happen that we can increase expressivity but retain the same complexity.

In the context of the scenarios introduced earlier, OWL DL fulfills the requirement with regards to decidability, but both decidable OWL languages (OWL DL and OWL Lite) have an exponential (or worse) complexity (Horrocks and Patel-Schneider 2004), which makes it possibly unsuitable for our use cases – since we have to expect to deal with a high number of instances. Languages that allow the use of algorithms that can be implemented with a tractable complexity are considered more suitable in cases where we can expect such a high number of instances as in enterprise settings. OWL2 introduces language profiles (Motik et al. 2008), which are well-defined subsets of the OWL2 constructs. These profiles have specific properties that are also guaranteed for all models adhering to these profiles.

Other aspects not mentioned in this classification, but relevant when describing an ontology, or every other knowledge structure, are covered by so-called meta-ontologies and metadata schemes thereof. Metadata schemes such as OMV (Hartmann et al. 2005) cover general information about ontologies, such as the size in terms of specific types of ontological primitives, the domain described, the usage scenarios, the support software and techniques, and so on. Many of these aspects are interrelated and can be traded against each other, as it will be elaborated later in this section. Potential developers and users of ontologies should be made aware of the trade-offs associated to engineering and using a particular type of knowledge structure. More precisely, these tasks require specific expertise, software and infrastructure, as well as the compliance with processes and methodologies, all under circumstances related to considerable costs.

These trade-offs are summarized in Table 3.1.

The considerations presented in Table 3.1 can be used as general guidelines to be taken into account and applied in the process of engineering an ontology. Their operationalization has to rely on methods which allow a quantification of costs and

[1] http://www.omg.org/technology/documents/modeling_spec_catalog.htm

Table 3.1 General trade-offs

Formality	A formal ontology is useful in areas which require sophisticated processing of background knowledge and automatic inferencing. This assumes the availability of mature tooling for these tasks. In addition, the more formal an ontology should be the higher the level of expertise and the costs of the ontology development processes. Finally, heavyweight ontologies can not be acquired automatically, as properties and axioms can not be feasibly learned from unstructured knowledge structures using the present software.
Shareability	The main advantage of a shared ontology is its capability to enable interoperability at the data and interoperability levels. Developing a commonly agreed ontology implies, however, additional overhead in terms of the development process to be followed, including methodological support and software to support the discussions and consensus reaching task. In addition, a shared ontology will not be able to optimally match very specific needs of many usage scenarios in which it is involved. Thus additional overhead to understand and adapt is required.
Domain and scope	First there is the aforementioned trade-off between the scope and the reusability of particular categories of ontologies. In addition, higher-level ontologies tend to be more costly, as they require specific expertise. The same applies for ontologies dealing with expert knowledge, such as those in areas of chip design. The size of a knowledge artifact (expressed, let's say, in the number of concepts, properties, axioms and fixed instances) is an important factor to be aware of, not only because of the direct relationship to the development and maintenance costs, but also because of the difficulties associated with the processing of large artifacts by reasoners and alike. There is a trade-off between the domain coverage of an ontology and the additional effort required to build, revise and use it.
Representation language	Besides to the link to the formality dimension, the choice of a representation language has consequences with respect to the ways an ontology can be used in knowledge inferencing tasks and the extent to which particular aspects of the knowledge domain can or can not be captured by the ontology. In addition, formal, logics-based representation languages require specific expertise within the ontology development and maintenance team.

benefits involved and their analysis (see Chapter 'Using Cost-Benefit Information in Ontology Engineering Projects').

3.3 How are Enterprise Knowledge Structures Being Built

In this section, we first give a short overview of the wiki technology Semantic MediaWiki (SMW) (Krötzsch et al. 2007) as a flexible tool for dealing with enterprise knowledge structures. Then we describe in more detail selected aspects of knowledge structure editing, leveraging, and repair.

Social software as a tool for knowledge sharing and collaboration is gaining more and more relevance in the enterprise world (Drakos et al. 2009). This is especially true for so-called enterprise wikis, that, just as wikis in the public Web,

provide their advantages of low usage-barriers and direct benefits within a company intranet. However, the simple provision of a Wikipedia-like internal page does not guarantee acceptance by employees; such wiki software needs to be customized to the specificities of the corporate context.

SMW provides this customization by combining the complementary technologies of Web 2.0 and the Semantic Web (Ankolekar et al. 2007). It enhances the popular open-source wiki software MediaWiki (http://www.mediawiki.org/wiki/MediaWiki) with semantic capabilities. In addition its functionality can be enriched with general-purpose extensions developed by the community[2] as well as custom extensions tailored to the needs of specific enterprise scenarios. The usage of the Semantic Web standards RDF, RDFS and OWL, and of ontologies enables the realization of comprehensive knowledge-management solutions, which provide integrated means to formally describe the meaning and organization of the content and to retrieve, present and navigate information.

In the following, we describe how enterprise knowledge structures are collaboratively built, enriched, and exploited using SMW.

Creating Structured Information Information stored in SMW can be converted into machine-readable RDF. In other words, it is possible to have property-value pairs explicitly assigned to wiki pages. Such a property-value pair can indicate a named link (a so-called *object property*) to another page, e.g., 'locatedInCountry', or a typed attribute (a so-called *datatype property*), e.g., String 'hasTag', Date 'hasFoundingDate', and Number 'hasHeight'. Such *properties* can be freely inserted into a page via wiki syntax or forms. Enterprise knowledge structures can be defined through *categories* (so-called *classes*) of pages with certain properties and encoded as an ontology in RDFS and OWL. The resulting ontology can be automatically or manually applied to the wiki (Vrandečić and Krötzsch 2006), for instance, in the form of categories, pages, wiki templates, forms and properties.

Retrieving Information The availability of machine-processable information facilitates the realization of concept-based search, presentation and navigation features going beyond traditional keyword-based approaches. The user can issue structured queries, addressing certain properties of a page, e.g., the customer of a proposal. All pages belonging to a category having certain properties can be listed as an overview, including links to those pages, e.g., all products within a specific price range. Various result formats can be used, starting from simple tables to more advanced calendars, time lines, and maps. Through *facetted search* one can incrementally filter lists of pages via keywords and property-ranges. More complex, but still user-friendly, querying following similar patterns as the standard Semantic Web querying language SPARQL (http://www.w3.org/TR/rdf-sparql-query/) is supported as well. When the user enters a keyword, the system looks for connections between pages described with the keywords and lists those pages

[2] Openly available at http://www.mediawiki.org/wiki/Extension_Matrix (MediaWiki) and http://semantic-mediawiki.org/wiki/Help:Extensions (SMW)

(Haase et al. 2009) In SMW, these queries are possible through forms on special pages, but can also be embedded as so-called *inline queries* in single pages.

Integrating External Information External sources can be integrated and their content merged with existing enterprise knowledge structures. The results can be organized as new pages or properties, referenced from other pages, and visualized in new ways.

Enterprise knowledge is rarely represented in RDF, but there are many tools available that deal with such transformations from established formats and standards, most notably tabular ones. The same applies to online knowledge sources such as Freebase (http://www.freebase.com), other SMW installations or the Linked Open Data cloud, for which a growing number of Web services delivering RDF are available (http://www.linkedopenservices.org). Orthogonal to the translation to RDF is the question of how to map specific elements of the source knowledge structure into the wiki model. Simply creating a page for each element within an external source and copying the data into the wiki may prove suboptimal for subsequent data usage. In Sect. 3.3.2 we provide additional details on SMW's integration features.

Improving Information Quality SMW specifically targets scenarios where knowledge is created in a decentralized manner – be that by exposing and integrating external sources, or by supporting collaborative editing and interaction. In such scenarios information quality can quickly become a problem. A prominent dimension we discuss here is consistency, both with respect to the primary sources and with respect to the domain at hand. For the former, SMW adheres to a regime in which users may only refer to, and comment upon the primary sources from within the wiki, while changes may only be undertaken at the level of these sources. For the latter, one can use an inference service operating on the wiki knowledge base. Deduction methods on the enterprise knowledge structures can provide insights about the wrong usage of categories, pages, and properties (Vrandečić 2009a). Most such errors cannot be automatically repaired, but at least, made visible to the users or administrators. For example, if the imported data contains information about a proposal with customer *X* and a wiki page exists about *X*, which is not a member of the customer category, adding that page to the category can be automatically suggested to the administrator. In addition, visualizing information in a structured way may lead to the identification of missing and incorrect information, which applies to both genuine wiki content and content from external sources. Users may not directly correct the latter, but they can rate it, and comment on it for revision. In Sect. 3.3.2 we will discuss a number of simple measurements whose results can indicate specific quality issues.

Interplay with other Enterprise Tools To maximize its added value for knowledge workers, SMW should not be used in isolation from existing enterprise systems and workflows. This is enabled by the information integration functionality presented earlier, and by a number of additional features targeting application integration. The content of a semantic wiki can be exported as RDF, as well as many other structured data formats, e.g., JSON, vCard, and BibTeX. Results of queries can be monitored for new pages and modified properties, and published as

RSS feeds and send per e-mail. Using HTTP requests to the wiki, external applications such as office productivity tools can access, add, and modify pages and properties.

In the following sections we go into more details of how enterprise knowledge structures can be edited, leveraged, and repaired.

3.3.1 Building Knowledge Structures Manually

The SMW OntologyEditor is an extension of Semantic MediaWiki for developing and maintaining knowledge structures (so-called vocabularies). As such it inherits many of the features and the mode of operation of Semantic MediaWiki. It targets Semantic MediaWiki users, but it also provides a comfortable interface for people less experienced in using wikis, in particular the wiki syntax. In this section we will briefly introduce the functionality of this extension – a more detailed account is given in (Simperl et al. 2010).

Main Page The main page is the entry point of the SMW OntologyEditor (see Fig. 3.2). It contains the primary navigation structure and links to important pieces of functionality, including the creation of new vocabularies consisting of categories and properties, the integration of other knowledge structures, such as folksonomies and external vocabularies encoded in RDFS and OWL, and knowledge repair (see Sect. 3.3.2). In addition, the user is provided with a short introduction to the tool, important links, as well as an overview of the content of the current wiki installation, in terms of namespaces of the individual vocabularies and a tag cloud.

Vocabulary Creation To create a new vocabulary one can use the corresponding link in the primary navigation menu, which leads to a form (see Fig. 3.3). There the user can enter a vocabulary name and a description and add categories and properties. Once the vocabulary is created the user is presented with a vocabulary overview, which contains automatically added metadata such as Flickr images in addition to the information manually provided by the user and a link to the *Create Category Form* (see Fig. 3.4).

The *Create Category Form* includes a short explanation and a number of input fields. They are autocompletion-enabled, by which the user is presented with a list of entities with a similar name to the one she is about to type-in (see Fig. 3.4). The user can enter a name for the category, refer to an existing vocabulary, define sub- and super-categories, and add new and existing properties to the category. Subsequently the system displays the category overview page illustrated in Fig. 3.5, including a tag cloud to easily access category instances (so-called entities) as well as the most interesting images on Flickr related to the category. Categories and entities are visualized in a tree-like hierarchy, which can be altered by clicking on the *Edit* links which open pop-ups for inline editing (see Fig. 3.6).

Knowledge Repair The knowledge repair algorithms can be accessed as so-called SpecialPages in the wiki.

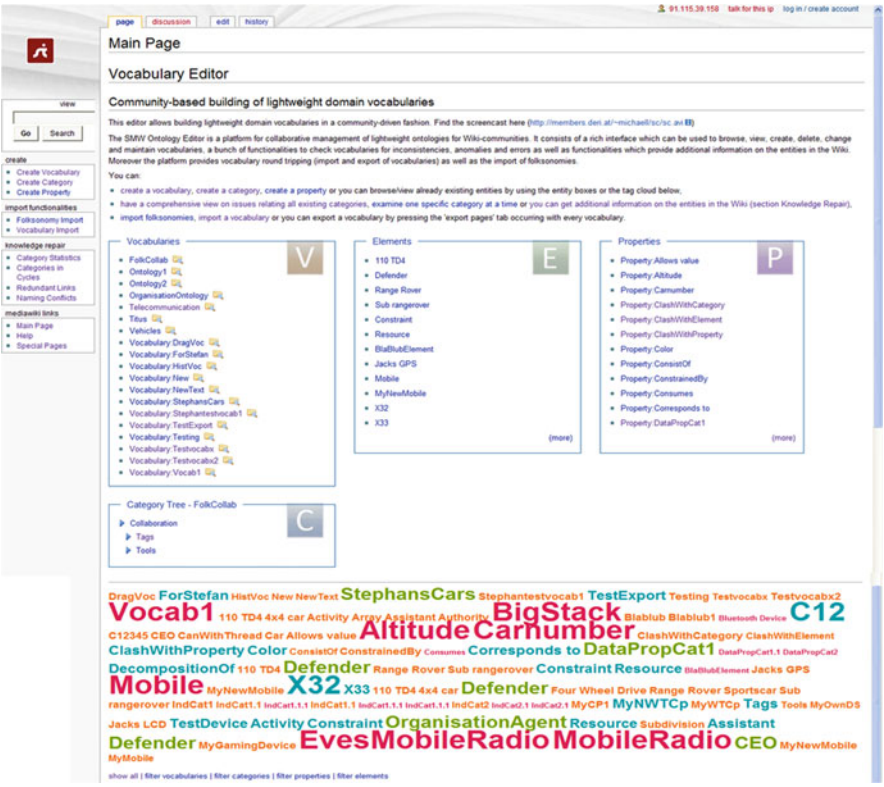

Fig. 3.2 Main page

After clicking on *Category Statistics* the system provides the user with a comprehensive overview of all categories available in the system, and potential modeling issues (see Fig. 3.11). At the top there is a table with explanations followed by a table with minimum, maximum and average values serving as basis for error detection. An additional table displays the corresponding values for all categories. Colors and symbols are used to direct the user focus to potential problems. If the user is not interested in all categories, but rather in a specific one she can click on the tab *repair* on the category page which will lead to an overview page of the specific category displaying similar information (see Fig. 3.10).

The SpecialPage *Categories in cycles*, shown in Fig. 3.9 lists cycles in a category hierarchy. The user then has to decide whether the specific cycle will be accepted or not. Redundancies in the hierarchy are displayed in the SpecialPage *Categories with redundant subcategory relations* (see Fig. 3.9). The user can decide which link is indeed redundant and delete it on-the-fly. The SpecialPage *Entities with similar names* provides the user with information about entities with similar names (Fig. 3.9). For each entity the system calculates the Levenshtein distance (Leveshtein 1966) to other categories and displays the results. In Sect. 3.3.2

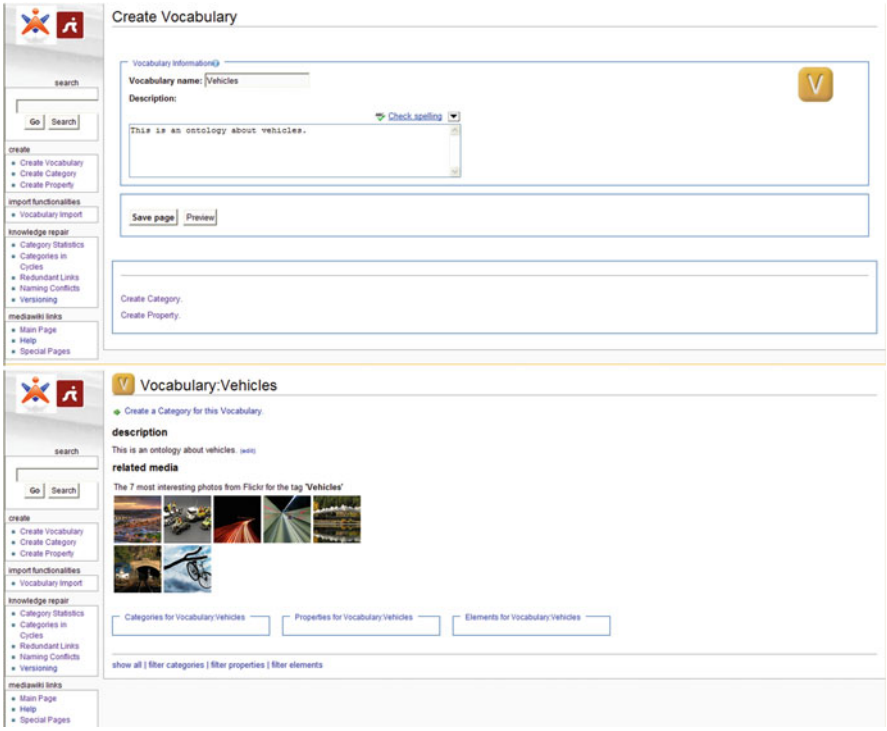

Fig. 3.3 Create vocabulary form and overview page

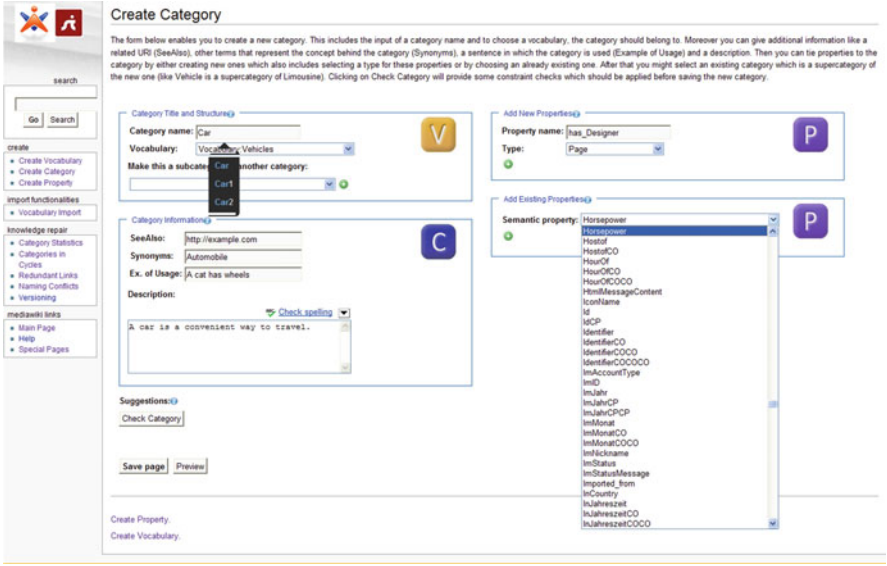

Fig. 3.4 Create category form

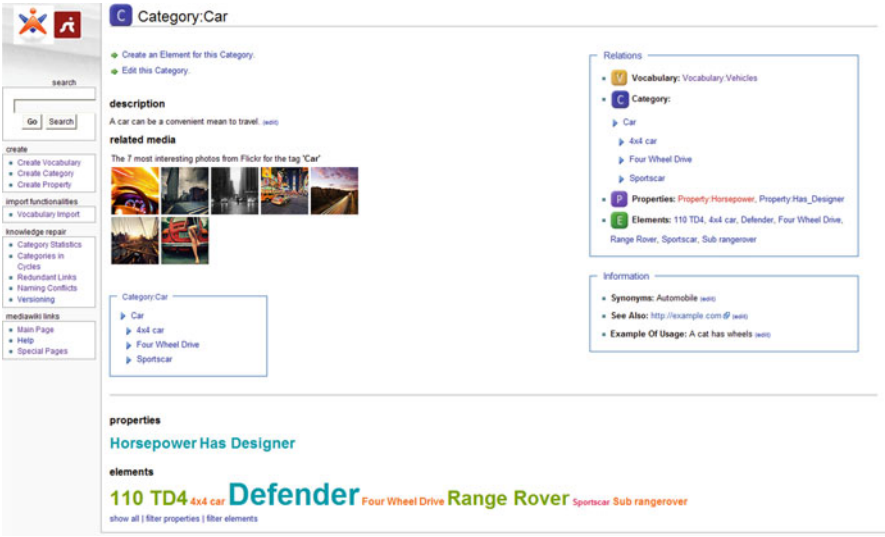

Fig. 3.5 Category overview

Fig. 3.6 Changing parameters via inline editing

we introduce additional knowledge repair features, such as the *Category Histogram*, the *Property Histogram*, *Categories with similar property sets* and *Unsubcategorized categories*.

Versioning The versioning SpecialPage gives an overview of the history of changes of vocabularies and categories. When a vocabulary or category is selected, a pop-up with detailed versioning information is displayed. On the left-hand side the user can choose between *Vocabulary Structure Changes* and *Category Changes*. Different versions are displayed (via AJAX) on the right-hand side of the pop-up. A selected version can be restored by clicking the *Restore Selected Version* button, as depicted in Fig. 3.7.

Import and export One of the advantages of Semantic MediaWiki as a knowledge management platform is its ability to provide integrated access to a multitude of knowledge structures, most prominently folksonomies and ontologies. The folkosonomy import relies on the technique described in Sect. 3.3.2 thus the

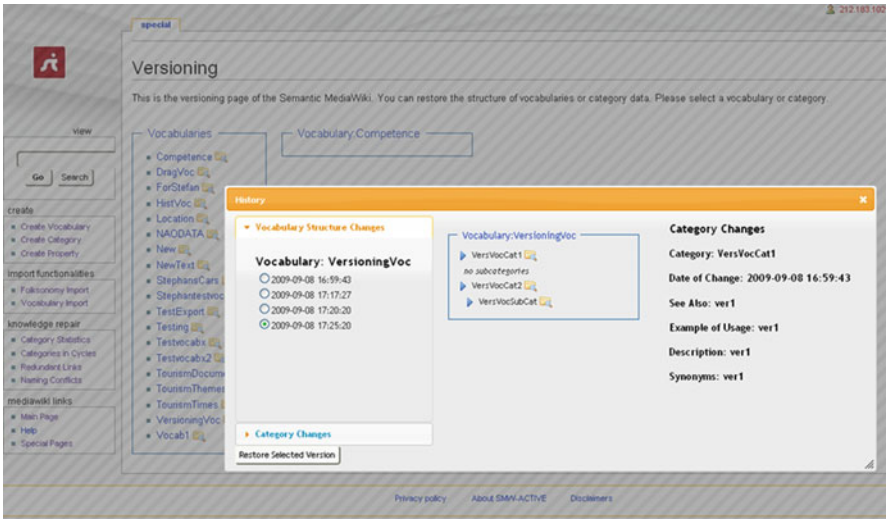

Fig. 3.7 Versioning

associated information – a collection of tagged resources such as bookmarks or conventional documents – has to be organized in a specific XML format. Given this, the folksonomy is enriched with additional structuring information and is transformed in a lightweight ontology which can be explored and further revised in the editor just as any other vocabulary. When importing an existing OWL ontology – for instance, one that was developed in a different ontology engineering environment – the system uploads the OWL file specified by the users, extracts all ontological entities and creates corresponding wiki content following the instructions defined in a so-called meta-model. This meta-model describes the types of ontological primitives supported by the editor – in this case, as we are dealing with lightweight knowledge structures, a subset of OWL consisting of classes, instances and properties, in particular specialization-generalization – and how they are mapped to SMW artifacts. Once this step is concluded, the resulting vocabulary can be further processed in a collaborative fashion in our tool. Every vocabulary can be locally stored as an OWL file using the export tab on the vocabulary overview page.

3.3.2 Leveraging External Knowledge Sources

Enterprise knowledge structures come in various forms, from database tables, standardized taxonomies and loosely defined folksonomies to strictly organized knowledge bases. To optimally support knowledge management tasks in a corporate environment Semantic MediaWiki needs to provide mechanisms to access,

integrate and use all these different formats. This is important for its acceptance as a knowledge management solution – as it builds upon established resources and platforms – and for its efficient use – as reusing existing resources can reduce costs and improve the quality of the resulting enterprise knowledge structures. In the previous section we have explained how such knowledge structures can be manually created and maintained. The techniques introduced in the following are complementary to this functionality. The first one adds a critical mass of formal semantics to folksonomies in order to overcome some of their typical limitations, such as the usage of abbreviations and alternative spelling, synonyms and different natural languages to tag the same resource. The resulting lightweight ontology can be explored, further developed and used in Semantic MediaWiki. In contrast, the focus of the second technique is on leveraging existing knowledge bases, which might contain significant amounts of (instance) data which could be useful within SMW. The implementation is based on Freebase as one of most visible collections of structured knowledge created in recent years; however, the mediator-based approach underlying the implementation can be equally applied to other knowledge bases.

3.3.2.1 Turning Folksonomies into Ontologies

This section gives an overview of our approach to extract lightweight ontologies from folksonomies. The approach consists of 12 steps that have to be carried out in the given order (see Table 3.2).

Step 1: Filter Irrelevant Tags In the first step, we eliminate tags, which do not improve the information content, but have a downgrading effect on the quality of the data basis. Unusual tags, which do not start with a letter are therefore filtered out. Additionally, uncommon tags are dismissed. In this context a certain tag is uncommon, if it is used less than a predefined threshold.

Step 2: Group Tags Using Levenshtein Metric The process of annotating a certain resource with tags is an uncontrolled operation, which means that no spell-checking or any other input verification can be assumed to take place. As a consequence typing errors, mixing of plural and singular forms, annotations in different languages and other possible minor discrepancies between tags are likely to occur. The Levenshtein similarity metric (Leveshtein 1966) is used to discover morphologically similar tags.

Step 3: Enrich Tags with Wordnet Wordnet is a rich resource of lexical information (Fellbaum 1998). The database is organized in so-called synsets and can be accessed locally or remotely over a simple user interface. If a certain tag can be found in Wordnet, one can expect the tag to be a valid English term. All tags which are covered by Wordnet are assigned a flag containing the exact number of occurrences in Wordnet synsets.

Step 4: Enrich Tags with Wikipedia Wikipedia is a large, high-quality, and up-to-date online encyclopedia. If a certain tag can be mapped to a Wikipedia article, this tag can be considered a correct natural language term. In addition, we can

Table 3.2 The 12 steps of our method

Step	Title	Description
1	Filter irrelevant tags	Consider only tag data that is shared between a sufficiently high number of users to increase the community representativeness of the prospected ontology.
2	Group tags using Levenshtein metric	Compare relevant tags using the Levenshtein similarity metric and group the highly similar ones. Tags within the same group are considered to have equivalent meaning and differences are assumed to be the result of spelling mistakes.
3	Enrich tags with Wordnet information	Check whether a tag is covered by the Wordnet thesaurus, which we consider a feasible indicator for a valid English term.
4	Enrich tags with Wikipedia information	Use information available on Wikipedia to enrich the tags.
5	Spell-check and translate	Perform English spell-checking and translate those tags that were neither found in Wordnet, nor in Wikipedia from foreign languages.
6	Update group assignments	Update the tag groups created in step 2 based on the additional information gathered in steps 3–5.
7	Find representative for each group	Select representative for each tag group based on its quality.
8	Create co-occurrence matrix	Create symmetric square matrix containing information on the frequency with which two tags (or tag groups, respectively) were used to annotate the same resource.
9	Calculate similarities	Apply vector-based algorithm (Pearson correlation coefficient) in order to detect similarities between vectors in the co-occurrence matrix.
10	Enrich co-occurrence matrix with co-actoring information	Augment co-occurrence matrix with the information about the frequency with which two tags (or tag groups, respectively) were used by the same author.
11	Create clusters	Create clusters of tags (or tag groups, respectively) on the basis of the calculated correlation coefficients and co-actoring information.
12	Create ontologies	Transform the tag clusters created in step 11 into SKOS ontologies exploiting all information gathered in the previous steps.

benefit from the redirect pages functionality implemented in Wikipedia, so that even when a tag is incorrectly spelled or abbreviated, there is a high chance to find the correct corresponding Wikipedia article.

Step 5: Spell-check and Translate Spell checking and translating single words (not sentences or pieces of text) can be done automatically at a high precision. We apply this additional step because after the Levenshtein similarity check and the exploitation of Wikipedia redirects, not all tags can be related to these resources. This might occur when a tag is misspelled, or a tag is not in English.

Step 6*: Update Group Assignments* In this step we update the tag groups defined in step 2 based on the information collected in steps 3–5 and eventually decide which tags are relevant for the ontology to be created. The step can be further divided into 3 activities: (1) the re-grouping based on spell-checking and translation results; (2) the re-grouping based on Wikipedia results; and (3) the selection of relevant tags.

Re-grouping based on spell-checking and translation results The first group update is triggered by the mapping defined according to spell-checking and translation results. In order to ensure consistent groups after this update, four different scenarios for mappings of the type `tagA -> tagB` have to be considered:

1. Neither `tagA` nor `tagB` are assigned to a group: in this case `tagA` and `tabB`, plus all other tags mapped to any of them, form a new group.
2. `tagA` is assigned to a group, but `tagB` is not: in this case `tagB`, and all other tags mapped to it, are included into the group of `tagA`.
3. `tagB` is assigned to a group, but `tagA` is not: just as in the previous case, `tagA`, and all other tags mapped to it, are included into the group of `tagB`.
4. Both tags are already assigned to the same or different groups: in addition to the group updates, those tags, which are already assigned to one of the corresponding groups, have to be considered as well. Existing group members of `tagA` will be assigned to the group of `tagB`.

Re-grouping based on Wikipedia results The second group update is performed if two tags are assigned to the same Wikipedia article. Just as in the previous update step based on spell-checking and translation results, we consider existing groups and its members, which means that also other group members may be affected by this update operation.

The selection of relevant tags We assume that all tags, or groups of tags, containing either a Wikipedia or a Wordnet reference, are relevant for the generation of ontologies. The relevancy of the remaining tags and groups thereof is based on their frequency of occurrence in the folksonomy, i.e., on their usage. If this frequency is below a certain threshold, the tag or the tag group will not be considered. All affected tags will, therefore, be marked with a corresponding flag, indicating that the tag is not relevant for future steps towards the generation of ontologies. If the frequency of usage is above the given threshold, the tag, or tag group, will be considered to describe a new term created by the tagging community.

Step 7*: Find Representative for Each Group* This step is about finding the most representative tag in a tag group. This decision is taken as follows: the tag groups defined in the previous steps can contain many single tags. By definition all tags in a group are equivalent to each other, regardless of whether they are misspelled, occur with a certain lower or higher frequency, or are translations from other natural languages. For the generation of ontologies, however, we need to identify which of these tags is the most representative for the meaning of the corresponding tag group. Preference is given to tags occurring in Wordnet and Wikipedia references, in this order. If neither is the case, the decision is based on the highest frequency of usage.

Step 8*: Create Co-occurrence Matrix* Co-occurrence matrices provide the means to derive some kind of semantic relation between two entities. Amongst many

others, this approach was chosen by (Begelman et al. 2006; Cattuto et al. 2007a, b; Simpson 2008; Specia and Motta 2007) to analyze connections between tag entities. The symmetric $n \times n$ co-occurrence matrix M contains information about how frequently two tag entities are used to annotate the same resource. The value m_{ij}, representing the intersection of ($entity_i$, $entity_j$) for $1 \leq i, j \leq n$, corresponds to the frequency with which the two tag-entities $entity_i$ and $entity_j$ were used to annotate the same resource. The diagonal elements m_{ij}, where $i = j$, of the matrix M contain information on how often the tag-entity $entity_i$ was used at all. This serves as a starting point for steps 9–11.

Step 9: Calculate Similarities The co-occurrence matrix is a starting point to derive relations between tag entities. From a simplistic point of view, the relation between co-occurrence values and the total frequency of tag entries (as proposed by (Begelman et al. 2006)) can be seen as a good indicator for the relation of two tag entities. This approach, however, has one important disadvantage: it does not take into account similarities of the two tags to other tags or tag groups. A vector-based similarity measurement, as proposed in (Specia and Motta 2007), resolves this issue. A vector represents a row (or column) of the co-occurrence matrix. The similarity measure is based in our case on the Pearson correlation coefficient. Algorithm 1 below shows how the Pearson correlation coefficient is calculated for two variables X and Y, the means $\bar{X}$ and $\bar{Y}$ and standard deviations S_x and S_y, respectively.

Algorithm 1 Pearson Correlation Coefficient

$$r = \frac{\sum_{i=1}^{n} (X_i - \bar{X}) * (Y_i - \bar{Y})}{(n-1) * S_x * S_y}$$

A positive coefficient value is evidence for a general tendency that large values of X are related to large values of Y and that small values of X are related to small values of Y. A correlation above 0.5 is an indicator that the two vectors are strongly correlated.

Step 10: Enrich the Co-occurrence Matrix with Co-actoring Information The outcomes of step 9 do not allow us to derive relations between tags. This holds in particular for tags that are used frequently, but only by a limited number of users. Usually the insertion of tags by spam robots is causing this phenomenon. Even though there are many related tags with correlation values below 0.5, the threshold can not be lowered any further without taking the risk to derive faulty relations as well. To cope with this issue we enrich with so-called "co-actoring information". This key-figure can be calculated in a manner similar to the co-occurence information, the only difference being the fact that the focus lies rather on the users instead of tags. As such, the co-actoring information for two tags is defined as the total number of users who used both tags.

Step 11: Create Clusters In this step, we aim at creating sets of strongly related tags that we refer to as "clusters". To do so we calculate the relation of a tag entity to the total number of usage and the co-occurrence/co-actoring information and raise the correlation coefficient if the relation proportions are high enough.

Algorithm 2 shows the exact formula, where *ccoff* denotes the correlation coefficient of two tag entities, #(*tag*1) and #(*tag*2), denote the total usage of a certain tag, *coac*(*tag*1,*tag*2) stands for the co-actoring information of the tag `tag1` and `tag2` and *cooc*(*tag*1,*tag*2) represents the co-occurrence value of the two tags.

Algorithm 2 Correlation Coefficient Strengthener

$$r = ccoeff * \left(\frac{coac(tag1,tag2)}{\#(tag1)+\#(tag2)-coac(tag1,tag2)}\right) * \left(\frac{cooc(tag1,tag2)}{\#(tag1)+\#(tag2)-cooc(tag1,tag2)}\right) * 100$$

The algorithm minimizes the problem of spam entries and related tags with lower correlation coefficients dramatically. Tag pairs with either a basis correlation above the defined threshold th_1 or with a strengthened correlation coefficient to reach the threshold are then automatically considered to be related and form the basis for a cluster.

Tags are merged into one cluster only if the calculated correlations between tag entities, which are indirectly connected by the transitive law, are above another threshold th_2. This means, only if $cooc(tag1,tag2) \geq th_1$, $cooc(tag2,tag3) \geq th_1$ and $cooc(tag1,tag3) \geq th_2$, the three tag entities belong to the same cluster. Additional tag entities are added to a cluster only if all correlation values, with respect to the other tags in the cluster, exceed the defined threshold th_2.

While useful, applying this strategy results in a relatively high number of very similar clustering differing only in one or two elements. To solve this issue we apply two smoothing heuristics as follows.

1. If one cluster is completely contained in another one, the smaller cluster is deleted.
2. If the differences between two clusters are within a small margin and, additionally, the number of elements of both clusters exceeds a certain percentage with respect to the total number of elements of both clusters, the smaller cluster is deleted and the tags not included in the larger one are added to it.

The second smoothing heuristics is depicted in Algorithm 3, where $\#(cl_1)$ and $\#(cl_2)$ denote the number of elements within the clusters cl_1 and cl_2, respectively. The relevant threshold in this algorithm is th_{cl}.

Algorithm 3 Second Smoothing Heuristics for Two Clusters

$$\begin{aligned}
&\text{if } \frac{\#(cl_1 \cap cl_2)}{\#(cl_1 \cup cl_2)} \leq th_{cl} \\
&\text{and } th_{cl} \geq \frac{\#(cl_1)}{\#(cl_1)+\#(cl_2)} \\
&\text{and } th_{cl} \geq \frac{\#(cl_2)}{\#(cl_1)+\#(cl_2)} \\
&\text{then } remove(cl_1), remove(cl_2) \\
&\text{and } insert(cl_1 \cap cl_2)
\end{aligned}$$

Step 12*: Create Ontologies* All the terms occurring in a cluster are assumed to be related to each other in some way. The concrete type of the inter-tag connections is, nevertheless, hardly resolvable. We consider this limitation to be of less importance for creating lightwight ontologies. We use the SKOS standard (http://www.w3.org/

2004/02/skos/), which allows establishing associative links between concepts without the need to further specify their semantics. More precisely, the SKOS property `skos:related` can be used to designate all kinds of relationships amongst terms within one cluster. The clusters themselves are considered to be the domain of the ontology, for which meta-properties (e.g., by using Dublin Core) can be included. In SKOS, `skos:ConceptScheme` is used to identify a certain ontology. As a consequence, all entities within this scheme have to include a reference to this scheme; this is achieved through the construct `skos:inScheme`. The terms within a cluster represent the entities the ontology consists of. This direct mapping is possible as within the SKOS language, there is no distinction between classes and instances. The construct to designate these entities is `skos:Concept` (Fig. 3.8).

The SKOS constructs previously mentioned allow us to define the basis structure of the ontologies. The information that was collected with respect to translations, spell-checks, and so on, is used to enrich the ontologies. The preferred label for a concept is the respective representative of a tag group, which is denoted by `skos:prefLabel`. If there are other terms within the same group of tags, which do occur in Wordnet, the corresponding term can be considered as a valid substitute for the preferred label, information which is captured through the `skos:altLabel` construct. As SKOS does allow language distinctions, this feature is

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:skos="http://www.w3.org/2004/02/skos/core#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dc="http://purl.org/dc/elements/1.1/">

  <skos:ConceptScheme rdf:about="http://www.folk2onto.com/250">
      <dc:description>{espresso, coffee}</dc:description>
      <dc:creator>Alex</dc:creator>
  </skos:ConceptScheme>

  <skos:Concept rdf:about="http://www.folk2onto.com/espresso">
      <skos:inScheme rdf:resource="http://www.folk2onto.com/250"/>
      <foaf:page rdf:resource="http://en.wikipedia.org/wiki/Espresso"/>
      <skos:prefLabel xml:lang="EN">espresso</skos:prefLabel>
      <skos:related rdf:resource="http://www.folk2onto.com/coffee"/>
  </skos:Concept>
  <skos:Concept rdf:about="http://www.folk2onto.com/coffee">
      <skos:inScheme rdf:resource="http://www.folk2onto.com/250"/>
      <foaf:page rdf:resource="http://en.wikipedia.org/wiki/Coffee"/>
      <skos:prefLabel xml:lang="EN">coffee</skos:prefLabel>
      <skos:hiddenLabel>coffe</skos:hiddenLabel>
      <skos:hiddenLabel>cofee</skos:hiddenLabel>
      <skos:prefLabel xml:lang="DE">kaffee</skos:prefLabel>
      <skos:related rdf:resource="http://www.folk2onto.com/espresso"/>
  </skos:Concept>

</rdf:RDF>
```

Fig. 3.8 Example ontology created through our method

also used for both preferred labels and alternative labels. If a translation was found for a tag, this information is attached to the label, otherwise the label is considered to be English. This is done by using standard XML annotation, e.g., `skos:prefLabel xml:lang = "EN"`. All other tags of a certain group are considered to be "hidden labels" for the corresponding concept. The set of labels marked by `skos:hiddenLabel` comprises common spelling mistakes.

3.3.2.2 Integrating Freebase into Semantic MediaWiki

This section gives a brief overview of an extension to SMW that allows the use of inline queries to query Freebase (http://www.freebase.com) content via a mediator. The mediator creates an MQL query (Metaweb Query Language, the query language used in Freebase), handles the communication with Freebase, and returns query results in the same way as for conventional SMW inline queries. A full documentation of the extension is available in (Ell 2009).

Imagine you want to create a list of all European countries and their populations within your SMW-based knowledge management system. This information is available in general-purpose knowledge bases such as Freebase, and can be imported into the local SMW installation. The query statement could look as follows, where the source argument is an extension of the original AskQL syntax indicating the external knowledge base to be used.

```
{{ #ask:[[ Category:Country]] [[ Located in::Europe]]
   | ?Population
   | source = freebase
}}
```

The AskQL query has to be translated into an MQL query, which could look as follows.

```
[{
   "/type/object/name" : null,
   "/location/statistical_region/population" :[{
     "number" : null
   }],
   "/type/object/type" : "/location/country",
   "/location/location/containedby" :[{
     "/type/object/name" : "Europe"
   }]
}]
```

In order to be able to perform this translation additional information is needed. In this case it is necessary to know that

1. the category *Country* maps to */location/country*,
2. the property *Located in* maps to */location/location/containedby*, and
3. the print request *Population* maps to */location/statistical_region/population* where the field storing the value has the name *number*.

The transformation, which essentially follows a local-as-view approach, is presented in detail in (Ell 2009). Mapping information is stored in pages via properties, thus being editable and reusable for various inline queries.

Category mapping information is stored on category pages using the property `freebase category mapping`. For example the page *Category:City* (the page describing this category in the category namespace) may contain the statement `[[ freebase category mapping::/location/citytown]]`.

Page mapping information is stored on pages in the main namespace using the property *freebase page mapping*. For example the page *Karlsruhe* may contain the statement `[[ freebase page mapping::#9202a8c04000641f800-00000000b283e]]`.

Property mapping information is stored on property pages using the properties *freebase property mapping* and *freebase property type*. For example the page *Property:Population* (the page describing this property in the property namespace) may contain the statements `[[ freebase property mapping::/location/statistical_region/population ;number]]` and `[[ freebase property type::number]]`. Path elements are separated by `';'`. If no type mapping is specified then the standard type *string* is assumed per default.

Print request mapping information is stored on property pages since print requests relate to properties. For storing the mapping information the property *freebase pr mapping* is used. For example the property page *Property:Located in* may contain the statement `[[ freebase pr mapping::/location/location/containedby]]`.

In case the mapping information is missing or can not be properly interpreted, the extension behaves as follows.

Ambiguities The page where mapping information is expected to be contained may contain the mapping property multiple times. For example a category page may contain several properties with the property name *freebase category mapping*. In this situation the mapping information is ambiguous and only the first result returned by the SMW database is used.

Property type If the property type of a property is not given using *freebase property type* then type *string* is assumed.

Page mapping information missing If no page mapping exists for page P then an MQL query is created where an entity is requested with name P. If the query is specified with parameter $language = L$ then an MQL query is created that requests an entity that has the name P in language L.

Category mapping information missing If no category mapping information is found then the category statement and all subordinated statements in the description object tree returned by the query processor are ignored.

Property mapping information missing If no property mapping information is found then the property statement and all subordinated statements in the description object tree returned by the query processor are ignored.

This behavior is robust since missing mapping information is ignored. In case of ambiguities or missing mapping information, a warning is displayed to the user. Thereby a step-by-step development and improvement of the query is supported.

3.3.3 Repairing Knowledge Structures

Quality issues are a natural consequence of the collaborative, integrated knowledge engineering approach followed by Semantic MediaWiki and its extensions. Therefore, our solution also includes techniques to support users in detecting and correcting potential modeling errors or missing information. This section provides an overview of the types of quality issues we deal with and the implementation of the associated knowledge repair functionality.

Similar Names In an ontology we have different types of entities. A common issue with adding entities to an ontology is that a user might overlook that the entity she intends to add is already in the ontology with a name slightly different from the name the user would have chosen. By adding the entity, the user introduces redundancy to the ontology which makes the ontology unnecessarily larger, and more error prone. To avoid such issues we measure similarities between entities via the Levenshtein distance, and present the results to the user, who then has to decide whether the entities under consideration represent the same and thus should be merged, or whether they do not represent the same and therefore should be kept separately in the ontology.

Similar Property Sets The idea here is to compare the property sets of ontology classes in order to identify potential similarities. The ontology editor introduced in Sect. 3.3 displays all the sibling categories which have at least 50% of their properties in common (see Fig. 3.12) for the user to decide for appropriate action.

Cycles and Redundancies This measurement identifies cycles within a specialization-generalization hierarchy (see Fig. 3.9). Similarly the knowledge repair functionality includes means to identify redundant is-a relationships, which are presented as decision support to the user.

Missing Properties The underlying rationale for this metric is the inherent difficulties experienced by knowledge modelers in distinguishing between the data and the schema level of an ontology. Here we display those ontological primitives that do not have any successors in the hierarchy, thus indicating missing specialization-generalization properties or misclassifications of specific entities as classes or instances.

Category knowledge repair The previously discussed attempts to solve problems are used primarily by certain users who aim at keeping the knowledge base consistent. The methods mentioned enable the user to get an idea which categories are part of a problem of the knowledge base no matter which taxonomy

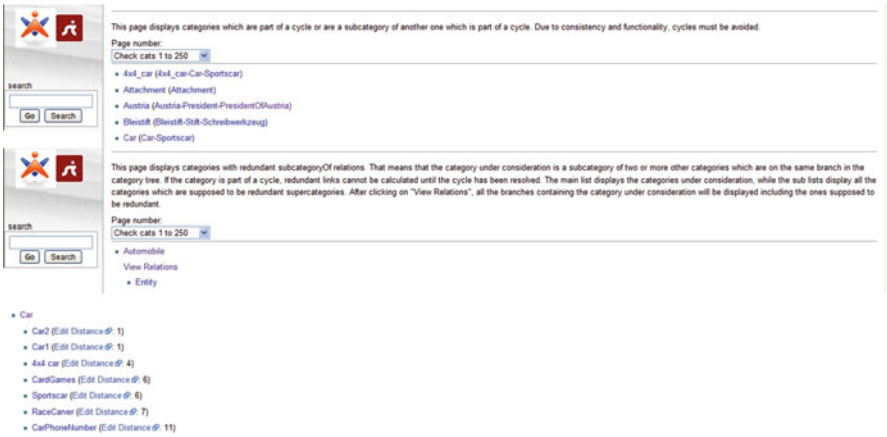

Fig. 3.9 Categories in cycles, categories with redundant relationships and entities with similar names

they belong to. However, there are also users who create an ontology because the domain under consideration is a domain of interest of such a user. Therefore the user might be keen on creating an error-free ontology. Instead of using each approach sequentially in order to resolve the issues about a certain category, the user also has the possibility to get all information about one category at a time. Besides the previously mentioned methods the user gets also information about minimum, average and maximum values which can be compared to the values of the category under consideration as well as information about the meaning of certain figures, which is useful for the non experienced users. In order to guide the attention of the user to severe problems these are marked with a symbol or red color. Minor problems are marked and all the other information is not marked (see Fig. 3.10).

Category statistics Some of the previously described methods provide the user with information of all categories regarding one specific type of problem. The method Category knowledge repair in contrast provides the user with information of all the types of problems regarding one specific category. This approach combines these two types of problem solving attempts. It displays all categories together with the results of each problem solving attempt. Therefore the user gets all information about issues regarding all categories. The use of this approach is to have a global view on the situation of taxonomies and the categories. Without such a comprehensive view it can be rather difficult to solve issues which spread over many categories. Then a user would have to jump from one category to another many times to resolve an issue. This gets more complicated if the branching of the category under consideration is more complex than it is with a category only having one supercategory and one subcategory. So far, the user gets quite the same information for all categories, as he does when using the Category knowledge repair approach for one specific category. In order to guide the attention of the

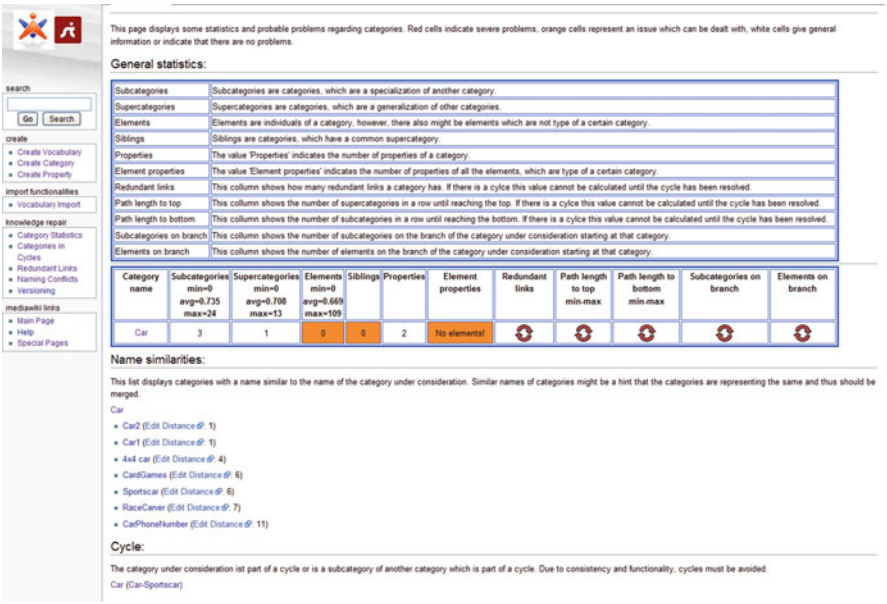

Fig. 3.10 Category knowledge repair

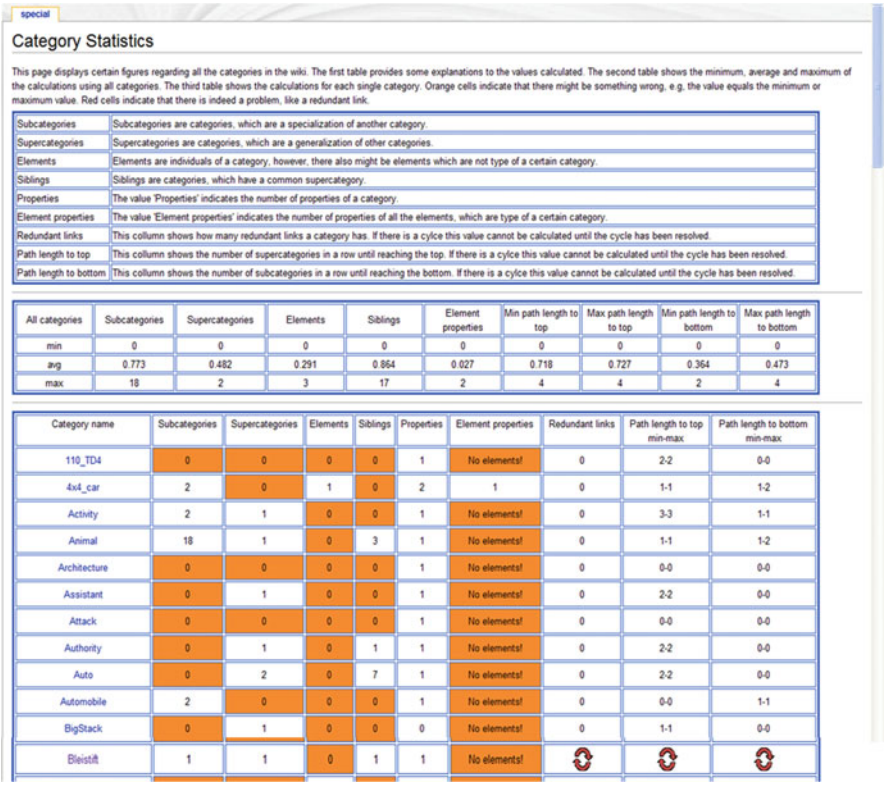

Fig. 3.11 Category statistics

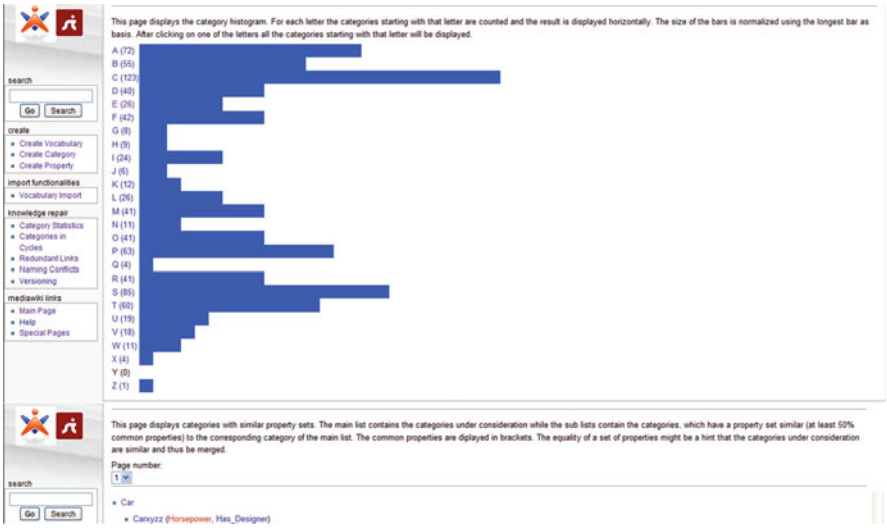

Fig. 3.12 Category histogram and categories with similar property sets

user to severe problems these are eye-catchingly marked. Minor problems are highlighted and all the other information is not marked as seen in the figure (see Fig. 3.11).

Category and Property Histogram In an ontology we have many entities starting with different letters. In order to get an overview of the distribution of entities starting with a specific letter in the ontology in relation to the alphabet a histogram can be very useful. It provides a comprehensive view on how many entities start with a specific letter in comparison to other letters (see Fig. 3.12). A normalized histogram can point out unusual things, however this requires that there is a certain number of entities in the database. The more entities there are the more likely they will follow a specific distribution regarding their first letters.

3.4 Conclusions

The chapter has covered the area of enterprise knowledge structures, starting from the requirements and research questions derived from use cases all the way to methodologies and implementations to bridge the different heterogenous structures that are in use today.

We expect that a common language for representing knowledge structures will foster further development and research in this area. The research results presented in this chapter are examples of what can be achieved once some foundational questions (such as the representation language or the necessary expressivity) have been settled, and we can move forward towards unifying knowledge management

tools and methodologies, further integrating results from heterogeneous areas in order to support the knowledge worker to the fullest possible extent.

Enterprise knowledge structures are heterogeneous in nature, and their integrated use requires a framework that allows understanding the trade-offs between different structures, and optimizes for given scenarios.

Many enterprises may already apply folksonomy-like systems. We have shown how folksonomies can be used as the foundation for developing lightweight ontologies which can then be in turn used to connect to further knowledge sources. Besides tagging, we have explored further Web 2.0 inspired paradigms, and implemented extensions to a wiki-based system that allows for the seamless integration of external data sources like Flickr or a company database. This system allows for explicit but lightweight management of an ontology within the wiki-interface, and powerful gardening and knowledge quality assessment tools.

References

Ankolekar A, Krötzsch M, Tran T, Vrandečić D (2007) The two cultures: mashing up web 2.0 and the semantic web. In: WWW '07: proceedings of the 16th international conference on world wide web, ACM Press, New York, pp 825–834, ISBN 9781595936547. doi: 10.1145/1242572.1242684, URL http://dx.doi.org/10.1145/1242572.1242684. 2007

Begelman G, Keller P, Smadja F (2006) Automated tag clustering: improving search and exploration in the tag space. In: Proceedings of the collaborative web tagging workshop co-located with the 15th international world wide web conference (WWW2006), 2006

Cattuto C, Loreto V, Pletronero L (2007a) Semiotic dynamics and collaborative tagging. Proc Nat Acad Sci U S A 104(5):1461

Cattuto C, Schmitz C, Baldassarri A, Servedio VDP, Loreto V, Hotho A, Grahl M, Stumme G (2007b) Network properties of folksonomies. AI Commun 20(4):245–262

Drakos N, Rozwell C, Bradley A, Mann J (2009) Magic quadrant for social software in the workplace. Gartner RAS core research note G00171792, Gartner. http://www.gartner.com/technology/media-products/reprints/microsoft/vol10/article4/article4.html. Accessed date Jan 2010

Ell B (2009) Integration of external data in semantic wikis. Master thesis, Hochschule, Mannheim, 2009

Fellbaum C (1998) WordNet: an electronic lexical database. MIT Press, Cambridge, MA

Fensel D (2001) Ontologies: silver bullet for knowledge management and electronic commerce. Springer, Berlin

Gangemi A, Guarino N, Masolo C, Oltramari A, Schneider L (2002) Sweetening ontologies with DOLCE. vol 2473 of Lecture notes in artificial intelligence (LNAI), Springer, Siguenza, Spain, pp 166–181, ISBN 3-540-44268-5

Gene Ontology Consortium (2000) Gene ontology: tool for the unification of biology. Nat Genet 25:25–30

Grau BC, Horrocks I, Motik B, Parsia B, Patel-Schneider P, Sattler U (2008) OWL 2: the next step for OWL. Web Semant Sci Serv Agent World Wide Web 6(4):309–322. ISSN 1570–8268. doi: http://dx.doi.org/10.1016/j.websem.2008.05.001

Gruber TR (1993) A translation approach to portable ontology specifications. Knowl Acquis 5 (2):199–220

Guarino N (1998) Formal ontology and information systems. In: Guarino N (ed) Proceedings of the first international conference on formal ontologies in information systems (FOIS), vol 46 of Frontiers in artificial intelligence and applications, IOS-Press, Trento, Italy, 1998

Haase P, Herzig DM, Musen M, Tran DT (2009) Semantic wiki search. In: 6th annual european semantic web conference, ESWC2009, vol 5554 of LNCS. Springer Verlag, Heraklion, Crete, Greece, pp 445–460, Juni 2009

Hartmann J, Sure Y, Haase P, Palma R, Suárez-Figueroa MC (2005) OMV – Ontology metadata vocabulary. In: Welty C (ed) Ontology patterns for the semantic web workshop, Galway, Ireland, 2005

Horrocks I, Patel-Schneider PF (2004) Reducing OWL entailment to description logic satisfiability. J Web Semant 1(4):7–26

Krötzsch M, Vrandečić D, Völkel M, Haller H, Studer R (2007) Semantic wikipedia. J Web Semant 5:251–261

Lenat DB (1995) CYC: a large-scale investment in knowledge infrastructure. Commun ACM 38 (11):33–38

Leveshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10:707–710

McGuiness DL (2003) Ontologies come of age. In: Fensel D, Hendler J, Lieberman H, Wahlster W (eds) Spinning the semantic web: bringing the world wide web to its full potential. MIT Press, Cambridge, MA

Motik B, Grau BC, Horrocks I, Wu Z, Fokoue A, Lutz C (2008) OWL2 web ontology language: profiles. W3C Working Draft 2 December 2008, Available at http://www.w3.org/TR/2008/WD-owl2-profiles-20081202/.

Pease A, Niles I, Li J (2002) The suggested upper merged ontology: a large ontology for the semantic web and its applications. In: Working notes of the AAAI-2002 workshop on ontologies and the semantic web, 2002

Simperl E, Wölger S, Bürger T, Siorpaes K, Han S-K, Luger M (2010) An ontology authoring tool for the enterprise 3.0. Taylor and Francis Publishing, 2010, London

Simpson E (2008) Clustering tags in enterprise and web folksonomies. Technical Report HPL-2008-18, HP Labs, 2008

Sowa JF (1995) Top-level ontological categories. International Journal of Human-Computer Studies 43(5/6):669–685. ISSN 1071–5819. doi: http://dx.doi.org/10.1006/ijhc.1995.1068

Specia L, Motta E (2007) Integrating folksonomies with the semantic web. In: Proceedings of the 4th European semantic web conference (ESWC2007), pp 624–639, 2007

Uschold M, Grueninger M (1996) Ontologies Principles, Methods and Applications. Knowledge Engineering Review 11(2):93–155

Vrandečić D (2009) Towards automatic content quality checks in semantic wikis. In: Social semantic web: where web 2.0 meets web 3.0, AAAI spring symposium, Springer, Stanford, CA, March 2009a

Vrandečić D (2009) Ontology evaluation. PhD thesis, Karlsruhe Institute for Technology, Germany, 2009b

Vrandečić D, Krötzsch M (2006) Reusing ontological background knowledge in semantic wikis. In: Völkel M, Schaffert S (eds) Proceedings of the first workshop on semantic wikis – from wiki to semantics, Workshop on Semantic Wikis. AIFB, ESWC2006, June 2006. URL http://www.aifb.uni-karlsruhe.de/Publikationen/showPublikation?publ_id=1211

http://www.springer.com/978-3-642-19509-9

Context and Semantics for Knowledge Management
Technologies for Personal Productivity
Warren, P.; Davies, J.; Simperl, E. (Eds.)
2011, XIV, 337 p., Hardcover
ISBN: 978-3-642-19509-9