Creation and Management of Structured Language Resources

PhD Thesis

Adam Rambousek
Creation and Management of Structured Language Resources

Abstract

In the past forty years, text processing by computer has developed from documents of a few thousand words to electronic databases of several billion words. For this reason among others, an effective system for compiling, storing, editing, accessing, and processing many different sorts of information about the language is of great practical importance.

Knowledge and language resources (e.g. ontologies, semantic networks, and dictionaries) not only serve end users, but also provide a valuable background information layer in the field of natural language processing or knowledge engineering. For example, ontologies and semantic networks are used in information retrieval applications, while dictionaries and lexical databases enhance word-sense disambiguation techniques.

This thesis analyses the current methods and tools for creation, management, and sharing of knowledge resources. Based on the analysis, a new lexicographic platform, named Dictionary Editor and Browser or DEB platform, has been designed and developed by the thesis author, taking into account all the features needed for resource creation and usability. In addition to the description of the platform itself, the thesis also presents details of nine software projects implemented by the thesis author during the past 8 years of the platform development, that are build using the platform, demonstrating various multi-disciplinary overlaps – e.g. filtering and combining big data resources, heavy multimedia use for the dictionary of sign language, or supporting multi-lingual knowledge detection system. All the presented projects have already been applied internationally in various research and study tasks solved by teams in more than 10 countries. Evaluation of the platform deployment and usage in practical applications and research projects is also provided in the thesis.
7 Selected projects

7.1 Family names dictionaries – combining big data resources ....................... 58
7.2 TeCU – terminological thesaurus methodology ......................................... 63
7.3 Dictionary of the Czech Sign Language ..................................................... 69
7.4 DEBVisDic – semantic network management ............................................. 74
7.5 Cornetto – complex lexical database ......................................................... 82
7.6 KYOTO – ontologies for knowledge detection ............................................ 87
7.7 PRALED – Czech lexical database .......................................................... 91
7.8 New Encyclopaedia of the Czech Language ............................................. 95
7.9 DEBWrite: prototype of “one click” dictionary .......................................... 97
7.10 Chapter conclusion ................................................................................... 98

8 Evaluation

8.1 Database storing optimizations ................................................................. 100
8.2 Linked Open Data rating .......................................................................... 101

9 Conclusion .................................................................................................. 103

References ..................................................................................................... 105

Appendix A Author’s publications

A.1 Book chapters .......................................................................................... 124
A.2 Journal papers ........................................................................................ 124
A.3 Peer reviewed conference papers ............................................................ 124
A.4 Other papers ............................................................................................ 126
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>DEB platform architecture.</td>
<td>46</td>
</tr>
<tr>
<td>6.2</td>
<td>Query execution time (in seconds) for a 3.6MB document</td>
<td>51</td>
</tr>
<tr>
<td>6.3</td>
<td>Query execution time (in seconds) for a 114MB document</td>
<td>52</td>
</tr>
<tr>
<td>6.4</td>
<td>Average time (in seconds) for the equality query</td>
<td>53</td>
</tr>
<tr>
<td>6.5</td>
<td>Average time (in seconds) for the full-text query</td>
<td>54</td>
</tr>
<tr>
<td>6.6</td>
<td>Average time (in seconds) for document update</td>
<td>55</td>
</tr>
<tr>
<td>7.1</td>
<td>FaNBI – editing entry, with virtual keyboard</td>
<td>59</td>
</tr>
<tr>
<td>7.2</td>
<td>Browsing the thesaurus, with detailed information for one term.</td>
<td>66</td>
</tr>
<tr>
<td>7.3</td>
<td>Entry relations visualized</td>
<td>67</td>
</tr>
<tr>
<td>7.4</td>
<td>Editing the term entry</td>
<td>67</td>
</tr>
<tr>
<td>7.5</td>
<td>Corpus evidence for usage of the selected term (teodolit)</td>
<td>68</td>
</tr>
<tr>
<td>7.6</td>
<td>Related words for the selected term (geodet)</td>
<td>69</td>
</tr>
<tr>
<td>7.7</td>
<td>Comparison of SignWriting (left) and HamNoSys transcription for the same sign.</td>
<td>70</td>
</tr>
<tr>
<td>7.8</td>
<td>Entry in the Czech Sign Language dictionary</td>
<td>71</td>
</tr>
<tr>
<td>7.9</td>
<td>Example of hand shape selection in &quot;iconic search&quot;</td>
<td>73</td>
</tr>
<tr>
<td>7.10</td>
<td>DEBVisDic – browsing three different wordnets (English, Greek, Korean).</td>
<td>75</td>
</tr>
<tr>
<td>7.11</td>
<td>DEBVisDic – synset preview</td>
<td>76</td>
</tr>
<tr>
<td>7.12</td>
<td>DEBVisDic – synset XML representation</td>
<td>77</td>
</tr>
<tr>
<td>7.13</td>
<td>DEBVisDic – comparison of tree and reverse tree view for synset gem:2, gem-stone:1, stone:4</td>
<td>77</td>
</tr>
<tr>
<td>7.14</td>
<td>Web interface for Global WordNet Grid with three interlinked wordnets.</td>
<td>79</td>
</tr>
<tr>
<td>7.15</td>
<td>Data collections in the Cornetto database</td>
<td>83</td>
</tr>
<tr>
<td>7.16</td>
<td>Cornetto Lexical Units, showing the preview and editing form.</td>
<td>84</td>
</tr>
<tr>
<td>7.17</td>
<td>Cornetto Synsets window, showing a preview and a hyperonymy tree.</td>
<td>85</td>
</tr>
<tr>
<td>7.18</td>
<td>Cornetto – context menu with inter-dictionary links</td>
<td>86</td>
</tr>
<tr>
<td>7.19</td>
<td>The schema of the KYOTO database within the KYOTO system</td>
<td>88</td>
</tr>
<tr>
<td>7.20</td>
<td>Linking wordnets, thesauri and ontologies within KYOTO database.</td>
<td>90</td>
</tr>
<tr>
<td>7.21</td>
<td>PRALED – the List window</td>
<td>92</td>
</tr>
<tr>
<td>7.22</td>
<td>PRALED – entry view with two editor windows for the headword jazyk (tongue)</td>
<td>93</td>
</tr>
<tr>
<td>7.23</td>
<td>PRALED – corpora concordances for an entry</td>
<td>94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1</td>
<td>Number of users that have signed the Licence of the NLPC DEB services at the Brno DEB server</td>
<td>100</td>
</tr>
<tr>
<td>8.2</td>
<td>Map of institutions with the DEB platform installed</td>
<td>101</td>
</tr>
<tr>
<td>8.3</td>
<td>Database evaluation, query execution time (in seconds) for a 1.8MB document</td>
<td>101</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Dictionary writing systems comparison ............................................ 22
5.2 Dictionary management, comparison ............................................. 23
5.3 User management, comparison .................................................... 23
5.4 Dictionary management, comparison ............................................. 24
5.5 Database engine and searching, comparison .................................. 25
5.6 Output formats, comparison ........................................................ 25
5.7 User interface, comparison .......................................................... 26
5.8 Helper tools, comparison ............................................................. 27
6.1 Dictionary writing systems comparison with DEB ......................... 36
6.2 Dictionary management, comparison with DEB ............................ 37
6.3 User management, comparison with DEB ..................................... 38
6.4 Dictionary management, comparison with DEB ............................ 38
6.5 Database engine and searching, comparison with DEB ................. 39
6.6 Output formats, comparison with DEB ......................................... 40
6.7 User interface, comparison with DEB .......................................... 41
6.8 Helper tools, comparison with DEB ............................................. 42
7.1 Original record from the IGI database and form included into FaNBI. . 62
7.2 Original Fiants record and form converted to include in FaNBI. .... 63
7.3 Thesaurus size statistics ............................................................... 65
7.4 Czech Sign Language Dictionary statistics .................................... 74
7.5 Translating API call – translate *bosque* (forest) from Spanish to Japanese . 89
7.6 PRALED – part of speech statistics ............................................. 95
I would like to give thanks and appreciation to my supervisor Aleš Horák and my former supervisor Karel Pala. Many thanks belong to all my colleagues at the NLP Centre for an amazing work environment and to all research colleagues working with the DEB platform for the great professional experience and inspiring comments. I also wish to express gratitude to my family, friends, and wolf proofreaders for their support and patience.
Knowledge sharing is no doubt very important for advancement of humanity, especially language resources are not only support better communication, but also help in understanding our language and try to describe the world. The oldest known extant structured language resource are *Ebla tablets* – clay tablets with the recording of Sumerian and Eblaite languages and first ever bilingual dictionary, dated between ca. 2500 BC and 2250 BC¹ (see [Gor82, Wel81]). Another well-known example of historic multi-lingual resource is the Rosetta Stone, dated 196 BC, a stele inscribed with a decree presented in Greek, Demotic and hieroglyphic scripts² (see [Par05]). Ontologies as a way for describing the nature and categories of beings traditionally dates back to Aristotle, however in a sense of structured information science resource appears in 17th century.

Structured language resources (e.g. ontologies, semantic networks, or dictionaries) not only serve to end users, but also provide valuable background information layer in the field of natural language processing or knowledge engineering. For example, ontologies and lexicons are used in the machine translation field since 1950s [Wea55]. Other tasks that may be enhanced by language resources include information retrieval applications, word-sense disambiguation, natural language generation, or question answering.

Some language resources were designed with the software processing already in mind (e.g.

---

¹Digital edition of the tablets is available at [http://virgo.unive.it/eblaonline](http://virgo.unive.it/eblaonline).

²The British Museum provides digitalized copy and many resources at [http://www.britishmuseum.org/explore/highlights/highlight_objects/aes/t/the_rosetta_stone.aspx](http://www.britishmuseum.org/explore/highlights/highlight_objects/aes/t/the_rosetta_stone.aspx).
lexical databases or formal ontologies [PHK10]), on the other hand dictionaries were mostly following the traditional method of printed books aimed at human reader. However, in recent years, with the expansion of Internet and almost constant online connection, the traditional paradigms of dictionary compiling and publishing are undergoing significant changes. Although Leffa found in 1992 that electronic dictionary allowed students to better understand studied foreign language [Lef92], and De Schryver in 2003 mentioned 118 possible advantages of electronic dictionaries over paper dictionaries [dSo3], the changes are taking place slowly and only recently. Tarp in [Tar12, p. 107] summarizes the consensus of many linguists and lexicographers:

There is little doubt that printed dictionaries will be published for a long period ahead but, at the same time, it is no secret that the electronic medium is gaining still more ground and will gradually overtake and outshine paper as the preferred platform...

New ways to publish and access to language resources also open new challenges, one of the main challenges for resource authors and application developers is "to find a trade-off between the huge amounts of data that are being made available to users and the effort users need to expend to find the information that is relevant to them." [VP12, p. 161] The software system and related methodologies developed by the author and described in this thesis are aiming to help in resolving these challenges.

Thesis outline

Chapter 2 describes most widespread types of knowledge and languages resources. Chapter 3 presents the formal representations and standardized models for encoding of various language resources. Chapter 4 explains the process of ontology and dictionary building, and the effects of software editing on the process.

The requirements for language resource editing software, together with the overview and comparison of existing software for dictionary, ontology and semantic network editing, is provided in chapter 5. This chapter also discusses current research in the field of user experience, effective data access, and linked data.

Chapter 6 is the main part of the thesis, describing the results of the author’s work. Section 6.1 describes the features of the developed DEB platform, comparing them with the previously described existing software. Next section presents the software architecture of the DEB platform, and the technologies used to build the system. Section 6.3 summarizes the reception of the DEB platform by the research community and its usage in international research projects. Chapter 7 describes selected projects and applications based on the DEB platform to demon-
strate the variability of the system and various interesting challenges that had to be solved. Finally, chapter 8 presents the results and evaluation of the DEB platform deployment.

Thesis contribution

The main goal of this work is to design methodologies and develop a software platform that will be freely available and to support the processes required for creation and management of structured language resources, including ontologies, dictionaries, semantic networks, or lexical databases.

This thesis describes the research results, namely the design of the dictionary writing platform, called DEB, comparing it with existing applications. One of the main advantages of the DEB platform over existing systems is the universality and adaptability for any type of language resource, while other applications deal with single resource type. The adaptability and variability is demonstrated by the description of selected notable research and lexicographic projects based on the DEB platform.

Both the DEB platform and applications presented in this thesis were designed and implemented by the thesis author.
The man is not wholly evil: he has a Thesaurus in his cabin.

J. M. Barrie (describing the antagonist in Peter Pan)

2 Knowledge resources

2.1 Ontologies

In computer science and informatics, ontologies are defined as the explicit formal specification of the terms and relations among them [Gru93, Gru95]. Ontologies are very important resource for sharing and analyzing domain knowledge. Most ontologies define individuals (instances, objects), classes (sets, concepts, types of objects), attributes (properties or features of classes or individuals), and relations between classes and individuals.

From historical perspective, the first attempt to build a scientific ontology was *An Essay towards a Real Character, and a Philosophical Language* by John Wilkins [Wil68]. He aimed to create universal language encoded in hierarchy of concepts, he also provided a guide how to represent words not present in his ontology (e.g. by synonyms, or periphrases). However, the resulting ontology is constructed rather confusingly and does not represent the knowledge in sound scientific manner. As argued by Borges "The classes and species that compose it are contradictory and vague" [Bor64], Eco "Wilkins never wondered whether other cultures might have organized the world after a different fashion" [Eco95], or Hanks "...be lumped together prescientific taxonomy and folk taxonomy" [Han13].

Much better classification of language concepts is provided by Peter Mark Roget in his *Thesaurus* [Rog52]. The Thesaurus is composed of six primary classes, each of them is composed of multiple divisions and sections, forming a tree of over a thousand branches for clusters of
words with the same semantic meaning. Thanks to the well-defined classification of terms, Thesaurus was used as one of the resources to enhance machine translation since the early days of research in the field (for example described by Masterman in [Mas57]). Although the original Thesaurus is more than 150 years old, it is still used as a lexical resource in natural language processing applications and research. See for example [JS01] describing the conversion of Thesaurus to complex electronic database and comparison with WordNet, or [TIC13] using Roget’s classification for text segmentation.

The ontologies are divided to three levels:

- **upper ontology** (top-level ontology, foundation ontology) – describing very general and abstract concepts that are shared across all knowledge domains,
- **middle-level ontology** – spanning multiple domains and serving as a bridge between upper ontology and individual domain ontologies (best known example is MILO [NP01]),
- **domain ontology** – representing concepts in one specific domain.

Well-known examples of upper ontologies include:

- Basic Formal Ontology (BFO) [Smi98] – small upper ontology designed for data integration and information retrieval in scientific domains,
- Cyc [LGP+90] – commercial ontology developed by Cycorp, smaller subsets were released as open source (OpenCyc) and for research (ResearchCyc),
- Suggested Upper Merged Ontology (SUMO) [NP03] – the largest freely available and open source formal ontology, created by IEEE,
- UMBEL (Upper Mapping and Binding Exchange Layer) [Str15] – subset extracted from OpenCyc, designed to provide hierarchy for linking and interoperability of different ontologies.

### 2.2 WordNet and semantic networks

WordNet [Fel98] is a semantic network for English language, created in the Cognitive science Laboratory of Princeton University. Parts of WordNet structure are modeled on the evidence of psycholinguistic experiments concerning the theories of human semantic memory and knowledge retrieval. Words (only in lexical categories of nouns, verbs, adjectives and adverbs) are grouped into sets of cognitive synonyms, called synsets. Current version of WordNet (3.1) contains 155,287 words organized in 117,659 synsets.

This thesis follows the common distinction of terms WordNet (the original Princeton WordNet) and wordnet (semantic network modeled similarly to WordNet, usually its extension or localization).

Each synset also contains a brief definition (gloss) and most of the synsets are extended with example sentences. All synsets are linked by semantic and lexical relations. The most frequently
used relation for nouns is *hyperonymy/hyponymy*, linking more general synsets to more specific synsets (e.g. *cat* is hyponym of *feline*). Another frequent relation is *meronymy/holonymy*, whole/part relation (for example *leg* is meronym of *table*). Hierarchy for verb synsets is expressed by the *hyperonymy/troponymy* relation, representing increasingly specific manner of verb (e.g. *whisper* is troponym of *talk*). Main relation for adjectives is *antonymy*, representing strong semantic relationship of "direct" antonyms.

Because the WordNet is organized in hyperonymy/hyponymy hierarchy, it is sometimes referred to and used as an ontology. However, there are several issues that need to be solved to use WordNet as a formal ontology. The issues were identified for example in OntoWordNet [GNV03] and OntoClean [OOG+02] projects, errors and heuristics for general semantic network building are also mentioned in [Ča12]. The most frequently mentioned problems with WordNet are confusion between concepts and instances, poor structure of the top ontology, or uneven distribution of domains [BFMP04].

WordNet has been utilized for many purposes in natural language processing, lexicography, or information science, including word sense disambiguation, automatic text generation, text classification, machine translation, or information retrieval.

WordNet inspired many related projects, like building semantic networks for various languages, extending the WordNet with more information, or linking WordNet data to other resources:

- *BabelNet* [NP10] – large multilingual semantic network integrating WordNet and Wikipedia,
- *BalkaNet* [Chr04] – wordnets for Bulgarian, Czech, Greek, Romanian, Turkish and Serbian,
- *DBPedia* [ABK+07] – extracting structured content from Wikipedia and linking to WordNet,
- *EuroWordNet* [Vos98] – wordnets for Dutch, Italian, Spanish, German, French, Czech, and Estonian,
- *eXtended WordNet* [MM01] – enhancing WordNet glosses with syntactical parsing, semantic tagging and transformation to logical forms,
- *ImageNet* [DDS+09] – image database organized according to the WordNet hierarchy,
- *SUMO* ontology providing mapping between WordNet synset and SUMO classes,
- *WOLF* [SF+08] – automatically constructed French wordnet.

### 2.3 Dictionaries

The most widespread type of knowledge resource is a dictionary. There are many attempts to provide exact and complete formal lexicographic definition, several of them are mentioned
by van Sterkenburg in [vSo3, chapter 1.1]. Hirst in [Hiro3] makes a meaningful distinction between "dictionary", which he defines as a lexicon intended for human users that contains linguistic knowledge, and "computational lexicon" which is designed for software processing. However, computational lexicon is a broad category, which may include ontologies, lexical database, or semantic networks, described separately in the following sections.

With the recent changes in the lexicography field mentioned in the introduction, we can see that the dictionary publishers and authors are trying to find new ways of using technology both for the benefit of users, and for themselves. The number of dictionaries available online for free is growing, although it seems that traditional publishers are slow to incorporate all the possibilities of the new media. This is discussed for example by Almind in [Alm05]:

Sadly, there are still publishers who believe that they can publish a proper internet dictionary by converting the digital form of a printed dictionary directly onto the internet and slapping a search engine on top of it. Such publishers usually work on the assumption that by showing off the printed design on screen potential customers will be compelled to buy the book. They will not. An internet dictionary is a work in its own right.

Prinsloo in [Pri12, p. 142] comes to a similar conclusion, concerned especially about lesser-resourced languages:

Modern technologies enable lexicographers to enhance the quality of electronic dictionaries and to incorporate a number of true electronic features to facilitate information retrieval in ways unthinkable in the paper dictionary dimension. However, to date EDs have not fully lived up to expectations.

In 2012, Macmillan Publishers announced that they would phase out printed version of their English Learner’s Dictionary and concentrate on the online dictionary. Michael Rundell, Editor-in-Chief of Macmillan Dictionaries, in [Run12] mentions the Grefenstette’s question "Will there be Lexicographers in the year 3000?” [Gre98], but changes it to "Will there be dictionaries?"

Similar argument was used by Tarp in [Tar08]: "Dictionaries are basically an answer to specific type of need...these needs are in principle not dependent on the existence of dictionaries...”

Lannoy in [Lan09] provides an analysis of readers’ interest in free online dictionaries and illustrates the need to adapt the existing dictionaries to modern technologies.

Dictionaries are currently in the transition from the traditional physical “product” to electronic “service” and publishers (both traditional, and newly emerging electronic-only) need to find new ways how to monetize their resources. This leads for example to providing API interface for integration of dictionary data into third-party applications, or sharing and com-
bining resources in standardized formats. And of course this brings up new requirements on the resource editing and management systems.

2.4 Lexical database

Lexical database (or lexicographic database) is a structured resource that contains as much information as possible regarding words and lexical units in a language. Lexical database is usually compiled from scratch based on the corpus evidence analysis. Lexicographers do not need to draw conclusions about the language, they just need to gather all relevant information (e.g. rich selection of example sentences, wide range of collocations, statistical information...). Well designed and built database may be reused for many kinds of dictionaries and also serve as a resource for natural language processing applications. For example, Atkins and Rundell in [AR08] describe the lexical database as a founding resource for dictionary writing (the process is detailed in chapter 4.3). Research use of lexical databases include for example a definition analyser by Alshawi [Als87] for automatical conversion of sense definitions into formal representations. Another example is the tool developed by Bird [Bir97] for automation of quantitative phonological research using the data from a lexical database.

There are cases of lexical databases compiled by merging existing dictionary entries, for example the German publisher Duden merged several separate dictionary resources into combined lexical database called *Wissensnetz deutsche Sprache* [Ale11]. Spohr in [Spo11] proposes to expand the lexical database model to multiple layers (called *pluri-monofunctional lexical database*) and use such database to provide customized dictionary information for each user.

Many lexical databases are available for various languages. Selection of recently published or prominent resources is described in the following paragraphs.

**DANTE** [AKR10] The English lexical database, providing detailed information for more than 40,000 most common words of English. As an information resource for the database, 1.7 billion word corpus was created. The GDEX tool [RHK+08] was used to heuristically detect the "best" example sentences from the corpus. The database is freely available for language research.

**OWID** [MS08] The compilation of several electronic lexicographic resources created at the Institute of the German Language. The database consists of about 300,000 short entries, and a fully detailed description for high-frequency words. All the resources are available online for free, together with non-dictionary resources (e.g. the bibliography of electronic lexicography).
Pralex [RK07] The Czech Lexical database created by the Institute of Czech Language (ICL) of the Academy of Sciences. The database was compiled using existing dictionaries by ICL, together with the data from the Czech National Corpus [ICN00]. The database contains 213,533 entries and is available for research, a small selection of entries is publicly available. Pralex database was built using the application described in chapter 7.7.

Cornetto [VMSvDVo8] Lexical semantic database of Dutch, combining wordnet-like semantic network (from the Dutch wordnet [Vos98]) with complex lexical database (Referentiebestand Nederlands [MMdM99]). Existing resources were aligned together, expanded with new data, and mapped to the ontology. The Cornetto lexical database was created (and later published) using the tools described in chapter 7.5. The database is publicly available, without the copyright parts from original resources.

Aralex [BMW10] A lexical database for Modern Standard Arabic, which is the pan-Arabic variety of Arabic shared throughout the Arab world and used for formal communication. Part of the Aralex building was the compilation of annotated Arabic corpus (with 40 million words). Combined lexical database contains information about orthographic forms, stems, roots, word patterns, and many statistical information about words. Aralex is freely available online.

2.5 Crowdsourced resources, Wiktionary

Following the success of Wikipedia in collaborative resource editing, many projects and applications emerged that make use of crowdsourcing. In linguistics and NLP research, crowdsourcing is usually used to manually annotate large datasets with semantic or syntactic information [Grá13], word sense disambiguation [Rum11], or to evaluate the results of automatic tools [Nev14], but may even help to detect epidemics outbreak [MGN+12].

The results of crowdsourcing experiments in NLP research were evaluated multiple times, concluding that combining annotation by several ”unskilled” annotators may result in cheaper and faster annotation. Snow et al. in [SOJNo8] evaluated annotation of five linguistic experiments by non-experts and conclude on the text analysis experiment:

Pooling judgments across all 7 tasks we find that on average it requires only 4 non-expert annotations per example to achieve the equivalent ITA\(^1\) as a single expert annotator.

Callison-Burch [CB09] experimented with evaluation of machine translation using crowdsourcing and suggests that:

\(^1\)inter-annotator agreement
By combining the judgments of many non-experts we are able to achieve the equivalent quality of experts.

In the field of lexicography, most prominent crowdsourced resource is the Wiktionary, sister project of Wikipedia. The goal of Wiktionary is to create a freely available “dictionary of all words in all languages” [Wiki15] edited by volunteers. Although the Wiktionary has started in 2002 and is constantly growing in size and number of languages covered, it has been studied and evaluated only in recent years. One of the first qualitative studies was accomplished by Fuertes-Olivera [FO09], who compared English and Spanish entries with expert-built lexicons and he proposed that Wiktionary need to enforce "strict editorial criteria in order to enhance their reliability and their users’ confidence."

Similar conclusion is suggested by Hanks in [Han12, p. 82]:

The hypertext structure [...] is eminently suitable as a model for the electronic dictionary of the future. What is needed is some way of ensuring that definitions are properly supported by links to corpus evidence...

Detailed analysis of Wiktionary structure, coverage, and quality in comparison with expert-built lexicons is presented by Meyer and Gurevych in [MG12]. They found out that overlap of lexical entries between Wiktionary and other dictionaries is quite small, thus making Wiktionary useful resource for additional linguistic information. However, they also note the issues with entry quality: "...it is a serious problem to distinguish well-crafted entries from those that need substantial revision by the community."

It is unreasonable to expect that the issues with entry quality will be resolved on large scale and in near future. All researchers considering to utilize the Wiktionary data need to remember that data quality may affect the results.

Wiktionary is already an enormous structured resource for linguistic research – containing almost 2 million word sense definitions for words from 1,500 languages (497,247 word senses for English). It has already been used in several research projects, including extraction of lexical knowledge [ZMG08a] or pronunciation [SOS10], computation of semantic relatedness of words [ZMG08b], or enhancing synonymy networks [NSG+09].

"Curiouser and curiouser!" cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English).

Lewis Carroll (Alice’s Adventures in Wonderland)

3

Formal models and standards

3.1 Ontologies

Ontologies are often represented using the Knowledge Interchange Format (KIF) [GF92] defined within the Ontolingua project [Gru93]. KIF allows a representation of arbitrary sentences in the first order predicate logic. First, representation formalism have to be implemented and this implementation is used to describe the ontology. During the Ontolingua project, common Frame Ontology was defined which can be used to represent ontologies. For example, SUMO [NP03] is encoded in KIF and exported to other formats automatically. Later, the interchange format was expanded to form the Common Logic framework, which was adopted as an ISO standard in 2007 [Del].

Formal languages for ontology description also include CycL, based on first-order logic and used in Cyc [LGP+ 90] project, or F-logic [KLW95], object-oriented and frame-based language.

Standardized language for ontology authoring is the Web Ontology Language [AH09] (OWL) published by the W3C, aiming mainly to be used in semantic web applications. OWL recognizes three basic concepts to form the ontology:

- class – a group of individuals that share the same properties,
- individual – instance of a class, properties may be used to relate one individual to another,
- property – binary relation that states a relationship between individuals or from an in-
The OWL specification includes three variants:

- **OWL Full** – designed as compatible with RDF Schema\cite{BG14}. Thanks to the design, all valid RDF documents are valid OWL Full ontologies. Main difference to other OWL variants is that classes may be instances or properties at the same time.
- **OWL DL** – based on description logic\cite{BLR03} and designed to provide maximum expressiveness while retaining computational completeness, and decidability.
- **OWL Lite** – designed as a simpler variant of OWL DL, based on the same formalism and principles, intended for encoding primarily classification hierarchy with simple constraints.

For encoding, OWL supports either abstract ontology syntax, or XML syntax for data exchange.

### 3.2 Dictionaries

Formal model of dictionary entries and lexical information is described in \cite{IKR00}. The authors present an abstract model not related to any existing encoding format, in order to avoid limitations or requirements of encoded representation.

Dictionary is defined as a recursive structure comprised of one or more nodes at each level. Any node $T$ can have attached property, either by an explicit assignment or inherited from a parent node. Property is described with predicate $PROP(T,P)$ (property $P$ is attached to node $T$). Properties are expressed as Feature–Value pairs and values are either atomic, or a set of Feature–Value pairs. Feature is described using predicate $FEAT(F,V)$ (feature $F$ has value $V$).

The model defines three types of features: cumulative features (feature value at child node is a combination of child and parent nodes features), overwriting features (if parent and child nodes define the same feature, only the child value is used), and local features (apply only at associated node and are not propagated through the structure). Feature types are described with predicates $CUM(F)$, $OVER(F)$ and $LOC(F)$, respectively.

Authors also define encoding of node structures to XML format, thus allowing conversion to other existing XML formats via XSLT \cite{Cla99}.

### 3.3 Text Encoding Initiative

The Text Encoding Initiative (TEI) is a consortium of many research institutions, libraries, and publishers, established in 1987 to develop, maintain, and promulgate hardware- and software-independent methods for encoding humanities data in electronic form \cite{Bur00}. One of the main results of the TEI consortium is the development and publication of Guidelines for Electronic
**Text Encoding and Interchange** (TEI Guidelines), that define a markup language to represent structural, renditional, and conceptual features of text documents. The first draft (P1) was published in 1990, the current version is P5, published in 2007 and regularly updated. Initially, the SGML markup language was used to encode documents. However, after the wide adoption of the XML standard, P4 version of TEI Guidelines published in 2002 switched to XML for document encoding.

TEI Guidelines are developed as a modular and extensible XML schema. The Guidelines cover a wide variety of text resources, ranging from books, dictionaries, or technical documentation, to language corpora, document alignment, graph networks, or morphological annotation. Although the scope of the Guidelines is very extensive, specific requirements may not be covered. To support document types not covered by the TEI Guidelines, it is possible to customize the schema and extend the document structure. On the other hand, if the schema is too complex for certain task (or for users to learn), the TEI consortium also publishes **TEI Lite** schema, subset of the schema that was designed to meet ”90% of the needs of 90% of the TEI user community” [Con14].

TEI Guidelines are not a formally approved standard, but the format is widely adopted for data encoding and sharing by many projects. Thanks to the well-defined XML structure, it is possible to convert between TEI and a variety of document formats.

### 3.4 Lexical Markup Framework

Lexical Markup Framework (LMF) is the ISO standard (24613:2008) [LMFo8] for natural language processing lexicons and machine readable dictionaries.

The goals of LMF are to provide a common model for the creation and use of lexical resources, to manage the exchange of data between and among these resources, and to enable the merging of large number of individual electronic resources to form extensive global electronic resources. [FBG+07]

The LMF model is structured in two parts: core package representing the basic information in a lexicon, and extension packages that define specific types of lexical resources (for example morphology, syntactic structures, or multi-word expression patterns). LMF lexicon structure is modeled using the UML diagrams. The standard provides the guidelines how to convert the UML model to the XML schema for lexicons. XML DTDs for the core and extension packages described in the standard are provided.

During the KYOTO project (see chapter 7.6), an extension package based on LMF was developed to represent the wordnet-like semantic networks, named WordNet-LMF [TMS+10]. The thesis author was one of the consultants during the WordNet-LMF development, and the
system described in the thesis was the first to support semantic network storing in the WordNet-LMF format. In this model, wordnet is split to three interlinked lexicons:

- lexical entries,
- synsets,
- relations between synsets (both in one wordnet, and between several wordnets).

However, during the later conversion of the German wordnet (GermaNet) [HF97] to the WordNet-LMF format for sharing, several issues were identified (due to more complex structure of GermaNet over Wordnet). Updates to the WordNet-LMF were proposed in [HH10].

### 3.5 WQuery

To deal with the issues encountered when querying the wordnet-like semantic networks, specific formal model was designed, named WQuery [Kubi12]. Wordnet is modeled as a graph of synsets, linked with relations hypernym and hyponym. Each synset has relations to its parts, modeled with a set of binary relations synset (linking word senses to synset), word (linking word senses to words), sensenum (linking word senses to sense numbers), pos (linking word senses to part of speech), and gloss (linking synsets to glosses).

The main goal of the WQuery model is to enable specific queries for wordnet data stored in the semantic network. Instead of defining new specific data model, it is more useful to adapt existing lexical formalism and develop specific query language on top of it. For example, LMF model is used to store many wordnet-like resources. Enhancement of an existing model also provides more options for knowledge resource sharing.
Creating resources

4.1 Computers in dictionary writing

Computer usage in dictionary writing started in the 1960s, however, at that time lexicographers still wrote entries on paper and computer specialists copied the texts into computer database. Longman Dictionary of Contemporary English published in 1978 introduced the restricted vocabulary of words in definitions and use of computers to check this requirement. Dictionary source and electronic edition data also included special grammar coding system and systematic semantic classification used extensively in natural language processing research [BB87]. The first edition of Collins English Dictionary (1979) [Col79] largely contributed to the development of computational lexicography by using computer databases for many editorial tasks.

COBUILD project, started in 1980, presented a new approach into computer utilization and built a large corpus of contemporary English. Relevant selections from the corpus were printed for the lexicographers. They still worked with paper, but the form for dictionary entry was structured to correspond with the fields in computer database [Sin87b]. The first edition of Collins COBUILD English Language Dictionary was published in 1987 [Sin87a].

During the 1990s dictionary publishers started to buy or develop in-house specialized dictionary writing systems and lexicographers were moving from papers to computers. Thanks to the development of computers and internet connection, language resources are available

\[\text{“A corpus is a collection of pieces of language text in electronic form, selected according to external criteria to represent, as far as possible, a language or language variety as a source of data for linguistic research.”} \text{[Sinos]}\]
on-line and dictionary writing systems support co-operation of geographically distant authors. This advancement is discussed for example by Rundell in [Run12, p. 25]:

...it is increasingly rare for dictionaries to be compiled and edited by permanent in-house teams... A far more common model is for lexicographers to work remotely, accessing both corpus and DWS through a fast Web connection...

The major revision project to publish completely revised third edition of the Oxford English Dictionary started in 2000 with expected completion in 2037 [Pat94, Bre07]. Unlike with printed versions, readers do not have to wait several years for each dictionary volume, but new material is published in the online version every three months [Sim14].

Also many software tools are now available that support dictionary writing, language analysis, or corpus querying, selected examples are described later in this thesis. Main goal of the lexicographic and language resource tools is not to completely automate the process of resource building, but to help authors with "boring" tasks and concentrate on interesting and more challenging matters.

...various software tools devised in recent years to streamline corpus-analysis and dictionary-creation process have had the effect of releasing lexicographers from the more banal tasks... and the editorial management to produce more systematic, more internally-consistent, and simply better descriptions of language in shorter time...[Run12, p. 29]

4.2 Ontology building

In recent years, ontologies are used not only in science and experimental projects, but thanks to W3C consortium and web-related standards also widely on the web – mostly in semantic web applications.

Ontology design and building process differs very much, based on the goals of the ontology. Noy and Hafner in [NH97] compare 10 specific ontologies in terms of purpose, coverage, size, formalism, and design process, concluding:

...there is great diversity in the way ontologies are designed and in the way they represent the world. Before real knowledge sharing and reuse will be practical, some standards should emerge in what an ontology should consist of, what the basic classes of object are that should be represented (for example, things, processes, relations), and how they are represented (not in terms of formalism but in terms of knowledge that should accompany the concepts).
Since then, several widely implemented standardised formalisms for ontology emerge and most ontology editing applications support sharing the data in RDF, OWL, or LMF formats (described in chapter 3).

Recently, the ontology building follows the guidelines described by Noy and McGuinness in [NM01].

The important first phase of the ontology development is the definition of ontology domain and scope. We need to consider intended future applications and users. Good way to determine the ontology scope is a short list of questions that a knowledge base using the ontology should answer, so called competency questions [GF+95]. Because many ontologies already exist, we need to consider if it is possible to reuse parts of existing ontologies.

In the second phase, it is useful to prepare a comprehensive list of terms related to the selected domain that will be included in the ontology or may be connected to the intended knowledge base.

The third phase consists of two closely interconnected tasks – definition of classes and class hierarchy, and definition of class properties. There are two approaches to class hierarchy development (see [UG96] for more details): top-down process starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts, and bottom-up process starts with the definition of the most specific classes with subsequent grouping of these classes into more general concepts. Sometimes, a combination of both processes is used. During the hierarchy development, class properties (or “slots”) are also described. Properties describe the internal structure of concepts, like the features and parts of objects, or relations to other classes.

In the last phase, individual instances of classes are created. Each instance is described by the class and set of properties.

As seen from the described ontology development process, almost all existing ontologies rely on the lexicographers’ intuitive knowledge of the language. Although many researchers have shown that corpus-driven linguistic and ontology resources are more reliable (for example in [Sin04, Han90, Dei99]), only a few ontologies are derived from corpus data. Most notable examples are Brandeis Shallow Ontology [PHR04] and CPA Ontology [HJ08].

4.3 Dictionary writing process

The most commonly used process for compiling monolingual dictionaries follows the guidelines as described by Hanks and Atkins in [Han10, Atk93, Mit03].

A very important background resource for the the dictionary building is a lexical database (described in chapter 2.4). Thus the first stage of the process is compiling or reusing the lexical database that contains the information about the language we want to describe.
An important decision for the dictionary editor is the macrostructure definition\(^2\), mainly the selection of the words or lexical units to include in the dictionary. This selection depends on the scope and target audience of the dictionary, see the example given by Hanks in \([\text{Han10}]\):

The editor of a dictionary for native speakers must aim at a very wide inclusion policy, for very often it is the rare and unusual words and senses that people will want to look up. The editor of a dictionary for foreign learners, on the other hand, will aim to be more selective, presenting and explaining just those words that, in his/her judgement, a foreign learner will need to know.

Dictionary writing software helps with the selection by presenting the information compiled from several resources, e.g. the lexical database, or the corpora.

In the next stage, lexicographers write the final entries. They need to extract the most relevant information for the intended users of the dictionary. This work must be done by experienced lexicographer who is familiar with carefully written user profile and style guide (instructions on content and presentation of dictionary entry) \([\text{Lan01}]\). To get a better understanding of the word meanings, lexicographers look for evidence of use in the corpus, which also provides valuable statistical information. Nowadays, very large corpora for many languages are available \([\text{JKK}^{+13}]\), allowing lexicographers to get a very good insight in language use. As summarized by Atkins and Rundell in \([\text{AR08}, \text{p. 61}]\):

*We don’t actually know how much data we need in order to account for a given linguistic feature...What we do know is that the more data we have, the more we learn.*

On the other hand, there is a danger in presenting the corpus data directly to the readers of the dictionary and a reason why corpus should mainly be used as a background evidence by lexicographers and presented only to skilled users. The biggest corpora are usually acquired from the web. And although the downloaded data are tidied up, it still contains texts from sources of varied quality, including texts with spelling errors or grammar errors. If such texts are presented directly to the readers in the final dictionary, readers may think it was written by the lexicographers. This point is mentioned for example by Asmussen in \([\text{Asm}]\):

...[users] might consider a concordance as further examples on the use of a certain meaning of a word or expression, but may be confused by the fact that they are often incomplete, may illustrate different meaning of the word or expression to the one in question, and may even show spelling mistakes or uses generally considered substandard...

\(^2\)Lexicographic macrostructure may be defined even more complex as discussed by Nielsen in \([\text{Nie90}]\), although the term usually refers to the list of lexical units.
When all the evidence is collected, the authors will write the meaning definitions. To keep consistency for all members of the same category or set (e.g. diseases, or months), pre-defined template entries are used.

In the case of bilingual dictionaries, the dictionary writing process is extended, as described by Atkins and Rundell in [AR08, p. 102]. The phase of "transfer" is added to the process. During this phase, experienced translators partially translate the lexical database to the target language. They do not have to translate every phrase and sentence, just suggest one or two translations for each word sense and necessary parts of non-general sentences and terms. In the final phase of writing entries, two lexicographers co-operate on each entry, speaker of source language and target language.

As mentioned in previous section, recently developed software tools take care of many tasks that had to be completed manually by the lexicographers, or provide more precise and statistically significant information on words and their use. Many examples are provided by Rundell and Kilgarriff in [RK11], concluding:

\begin{quote}
Automated lexicography is still some way off. In particular, we have not yet reached the point where definition writing and (hardest of all) word sense disambiguation (WSD) are carried out by machines.
\end{quote}
People like to be told what they already know. In short, what people think they want is news, but what they really crave is olds.

Terry Pratchett (The Truth)

5

Analysis of existing systems and current research

5.1 Dictionary Writing Systems

Michael Rundell and Sue Atkins in [AR08] define a dictionary writing system from the lexicographers’ point of view as a program “designed to manage the entire process of producing a dictionary, from compiling the first entry to outputting the final product for publication in print or electronic media”, composed of a text-editing interface, a database and a set of administrative tools.

As described by Kilgarriff in [dSo6], a dictionary writing system is “a piece of software for writing and producing a dictionary. It might include an editor, a database, a Web interface and various management tools...”. Abel in [Abe12, p. 85] draws more general definition of “a software program that is closely connected to a series of applications that together serve to assist dictionary production”.

As we can see, there is no clear, widely shared definition, although everyone agrees that a dictionary writing system is more complex than just the editor of a dictionary entry and supports all the steps in dictionary publishing process. Different requirements on a dictionary writing system are discussed in the following sections.

As with the definition, there is not just one precisely defined term for this type of application. Dictionary writing system is the most common term (for example in [ARo8]), and other terms
include dictionary management system, dictionary production software, lexicographic workbench, or dictionary editing system.

5.1.1 Existing dictionary writing systems

Since the beginning of the computer assisted dictionary editing, publishers usually developed their own in-house software for the task. Main reason for this decision was that no specialized dictionary software was available at first. Later, the publishers did not want to share their know-how or were afraid that a third-party software does not cover the internal process of dictionary creation efficiently. Examples of internally developed software include Pasadena for the Oxford English Dictionary, DicSy of Swedish publisher Norstedts, or ANW article editor by the Insituut voor Nederlandese Lexicology.

In recent years, the trend is changing in favour of generic dictionary writing systems created not by publishers, but by independent software developers. As with all technology outsourcing, there are several main reasons for the decision. The publisher does not need to employ software specialists and spend money and time on the development process. Usually, the licensing costs are much lower than in the case of in-house built software. For example de Schryver also argues that authors of generic dictionary systems spent many years perfecting the software based on their own experience and feedback by many users, that it is very hard to achieve such level of dictionary management features, user-friendliness, or enhanced functionality with a newly starting project [dSt1].

On the other hand, the potential users of the generic dictionary writing system have to take care of analysing their requirements for dictionary editing and management, compare the available software to choose the best option, and finally customise the selected software to meet the requirements.

Currently, there are four commercial applications available in active development and widely used that may be described as the generic dictionary writing systems. Following sections compare their features from several different perspectives.

IDM DPS The abbreviation was formerly explained as Dictionary Production System, however the system is now known as Digital Publishing System. The project is developed by IDM (France) and the notable uses include the creation of DANTE lexical database (see section 2.4) [IDM06].

iLEX Lexicographic software developed since 2004 by Erlandsen Media Publishing (Denmark). This software is developed in Java and is the only one in this comparison working on all major operating systems. In 2009, iLEX was adopted by Van Dale, leading publisher of Dutch dictionaries [Erlo4, dGM10].
TLex   Formerly known as TshwaneLex, this system is a part of lexicographic software suite (including also corpus tools, or terminology database). TLex is developed by TshwaneDJe (South Africa) and notable clients include Oxford University Press or Pearson/Longman publishing [JdSo4].

ABBYY Lingvo Content   ABBYY Lingvo is the electronic dictionary platform developed since 1989 by ABBYY (Russia). It started as a dictionary viewer and the viewer is still the main product of the platform. In 2003, dictionary editor ABBYY Lingvo Content was introduced as part of the platform. Currently, most of the dictionaries published through the Lingvo viewer are compiled using the Lingvo Content editor [Ryl10].

Section 6.1 describes how are the dictionary writing systems features implemented in the DEB platform developed during this thesis.

5.1.2 DWS overview

One of the first features to compare is the price of the software and supported operating systems. The authors and publishers usually have an IT infrastructure already, thus they need the software to fit with the infrastructure. If they are planning to buy new machines or operating systems for the project, the cost differences may still affect the choice of suitable dictionary writing system.

Support for multiple users is also an important feature to consider. If the dictionary writing system is to be used by the team of editors, client-server software is a much better choice (as discussed previously). On the other hand, if only one editor is working with the software, it is possible to save money by choosing the software accordingly.

Table 5.1: Dictionary writing systems comparison

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Windows</td>
<td>Win, Linux, MacOS</td>
<td>Win, MacOS</td>
<td>Windows</td>
</tr>
<tr>
<td>price</td>
<td>1000 €+</td>
<td>400–7000 €</td>
<td>150–1900 €</td>
<td>150 €+</td>
</tr>
<tr>
<td>support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>type</td>
<td>client-server</td>
<td>client-server, single-user</td>
<td>client-server, single-user</td>
<td>client-server</td>
</tr>
</tbody>
</table>

5.1.3 Dictionary management

When starting a new dictionary project, users need to specify the dictionary entry structure. All the dictionary writing systems offer a pre-defined entry structure that fits the need of a generic dictionary, based on the developers’ experience with previous works.
However, one setting does not fit all possible applications, thus a way to define a custom entry structure is essential. As discussed previously in section 3, XML format is the best option for storing dictionary entries. There are several ways to define the structure of an XML document, most widespread being the DTD or XML Schema [Fal01]. Because the right format definition may be difficult for complex entry structure, the definition is usually set-up by the technical support of the dictionary team at the beginning of the project. Some dictionary writing systems provide graphical interface to define the structure by end-users.

Larger dictionary projects may work with several dictionaries. For example, this feature is required when building different dictionaries from one source lexicographic database, or using several source databases to compile dictionary entries. It is preferable to support grouping the dictionaries as needed for each project.

Table 5.2: Dictionary management, comparison

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictionary structure</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>add new dictionary</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>structure definition: DTD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>- XML Schema</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>- edit in GUI</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

5.1.4 User management and access control

When a team of lexicographers works on a dictionary, the system needs to authorize the users and check their access permissions. Usually, users access permissions are based on their role in the project (e.g. editor, reviewer). It may be useful to reuse authentication from external accounts, for example operating system user accounts.

If the system is used to build several dictionaries, users may acquire different roles for each dictionary. Each project may need a different set of roles, ranging from basic distinction of read-only and write access to complex system for large teams with several levels of supervisors and editors.

Table 5.3: User management, comparison

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>user accounts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>external accounts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>user roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
5.1.5 Writing process management

The system provides facilities to manage the dictionary writing process. These facilities can also differ based on the project needs. In a small team, a mechanism to ensure that users do not overwrite each other’s edits while working on the same entry is enough. This is usually implemented as a sort of entry locking, when users select entries they are working on and nobody else can edit them in the meantime.

For bigger teams or projects with strict deadlines and rules, the system supports managing of the different phases of dictionary writing process. Dictionary editor assigns a set of entries to each lexicographer and determine the deadlines. Then the system may keep track of the progress and provide useful statistics. For example, if the dictionary size is limited, the system may check completed dictionary length and suggests entries to shorten.

When the entries are written, reviewers are assigned and their comments and fixes are recorded. Full editing history for each entry needs to be saved with a possibility to revert to previous stages.

The system also helps with a pre-publication process, for example, selecting the entries and materials to include in different dictionary versions (e.g. electronic or printed). Of course, the system needs to provide the dictionary data to the publisher in a negotiated format (compared in more detail in section 5.1.7).

<table>
<thead>
<tr>
<th>Table 5.4: Dictionary management, comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry locking</td>
</tr>
<tr>
<td>assign work</td>
</tr>
<tr>
<td>progress statistics</td>
</tr>
<tr>
<td>editing history</td>
</tr>
</tbody>
</table>

5.1.6 Database engine and searching

Systems that allow a free structure of entries, store the data directly in the XML format. There are basically two options to store XML documents. Either in the native XML database, or in an SQL database with XML support (“XML-enabled”). The choice of a particular database usually depends on the internal architecture of the dictionary writing system. Some systems are able to work with different databases and the users may select their preferred database. For comparison of native XML and XML-enabled databases, see paper [NLB 02]. For comparison of native XML databases and their performance to store dictionary data, see section 6.2.3.

All systems support basic pre-defined searches, e.g. for the value of headword. For complex search, users are able to select any element in the entry structure and specify its value, also com-
bining several search queries. The best way to implement complex search queries on the XML document is the support of XPath [CD99] and XQuery [BCF10] standards.

<table>
<thead>
<tr>
<th>Database engine and searching, comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>database</strong></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>stored as XML</td>
</tr>
<tr>
<td>complex search</td>
</tr>
<tr>
<td>regular expr.</td>
</tr>
<tr>
<td>XPath, XQuery</td>
</tr>
</tbody>
</table>

### 5.1.7 Output Formats

Regardless of internal entry structure, dictionary writing systems support data export in XML format for sharing with other users or applications. The systems also allow export to various output formats.

For electronic publishing, most common option is export of the dictionary entries to an HTML format and publish on website. For printed publication, the format is decided by the publisher, most widely supported format is PDF. However, if the publisher needs to make changes to the dictionary before publication (for example, fixing the graphical layout), editable format for the DTP software is a better choice.

Instead of common publishing formats, ABBYY Lingvo Content supports export to the DSL format that is used in the ABBYY Lingvo platform and allows publishing both in the connected applications and in print.

Because the data are usually saved in XML format, output is produced with the XSLT templates [Cla99]. Some systems contain graphical tool to set the templates without technical skills.

<table>
<thead>
<tr>
<th>Output formats, comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XML export</strong></td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>XSLT</td>
</tr>
<tr>
<td>GUI setting</td>
</tr>
<tr>
<td>electronic format</td>
</tr>
<tr>
<td>publishing format</td>
</tr>
</tbody>
</table>

25
5.1.8 User interface

Different users prefer different ways of editing entries and the dictionary writing systems usually support several modes.

The most common way to edit the entry is the form in “flat” structure that contains input boxes for all the entry parts. Another sort of visualization is a similar form, but the input boxes are arranged in a tree structure copying the structure of an XML entry. Some dictionary writing systems allow users to directly edit the preview of a dictionary entry.

Table 5.7: User interface, comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry form</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>tree form</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>editable preview</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>customizable</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>localization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

5.1.9 Helper tools

Dictionary writing systems include many tools to make the life easier for users. Most frequent tools include the following features.

The system is able to check if the entry definition contains only words from the limited group of words, so called defining dictionary. This feature is useful for compiling learner’s dictionary.

To save time when editing the entry, users may define and quickly insert repeated parts of an entry, quick template, both with entry structure and entry content.

In the dictionary, many entries are usually inter-connected, mentioning specific sense or part of the linked entry. The dictionary writing system keeps track of the connections and automatically updates the information when changed, this feature is called cross-reference checking.

With the growing popularity of electronic dictionaries, the software needs to support multimedia content. The dictionary writing system contains a library of multimedia, like images, and audio or video recordings, allowing users to add the media when compiling entries.

Currently, all the widespread dictionary writing systems use Unicode character set, thus it is possible to enter almost any letter in different character scripts (e.g. medieval Latin letters are missing). However, it is difficult to write “non-standard” characters with common keyboard setting, or combine characters from different scripts. It is helpful to include a way of entering different characters, usually in a form of a virtual keyboard.

As was discussed in section 4.3, corpus evidence is important for dictionary compiling, both in the preparation phase (e.g. for statistics) and when compiling individual dictionary entry.
Thus it is an advantage if the dictionary writing system is integrated with corpus and allows users to search for a word in a corpus and get the corresponding sentences.

Table 5.8: Helper tools, comparison

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
</tr>
</thead>
<tbody>
<tr>
<td>defining dict.</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>quick templates</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>cross-references</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>multimedia</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>virtual keyboard</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>corpus</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

5.2 Ontologies and Semantic Networks

5.2.1 Ontology Software

Although there are many ontology editing systems, most of them are not developed any longer. The three editors described in the following text are most widely used nowadays to build and edit ontologies.

The applications support common requirements for ontology editing. It is possible to create and edit the ontology, classes, objects, instances and various properties, both in textual and graphical form. Import and export of the common formats to save ontologies is implemented. An important feature is visualization and visual navigation within the ontology model.

**Protégé**  A set of tools that supports creation, visualization and manipulation of ontologies for use in knowledge applications developed by Stanford Centre for Biomedical Informatics Research [pro10].

**OWLGrEd**  An extension of Protégé that supports development and visualization of ontologies in the UML notation [FS00], developed by the University of Latvia [BBČ+10].

**NeOn Toolkit**  Ontology engineering environment that supports whole ontology engineering life-cycle, developed during the European project coordinated by The Open University [HLS08].

**Swoop**  An ontology editing software tailored specifically for OWL ontologies, developed as standalone Java application "that attempts to provide the look and feel of a browser-based application" [KPS+06].
5.2.2 Semantic network software

As discussed in section 2.2, most widely used semantic network is Princeton WordNet and its extended or national variants. The original WordNet is stored in text files edited by specialized tools, this format is still used in the latest PWN 3.0. Several applications are used for wordnet building and are described in the following texts.

**Polaris** An editing tool developed during the EuroWordnet project [Vos98]. Polaris tool supported import of wordnet fragments from text files and editing of synset entries. The software is no longer supported and is licensed by ELRA as part of EuroWordnet package. For wordnet browsing, the publicly available viewer Periscope was developed.

**VisDic** A generic semantic network editor that stores data in XML format was developed by NLP Centre Faculty of Informatics MU for the building of national wordnets in Balkanet project [PPo2, HS04]. A unified format to store wordnets was introduced together with VisDic. The editor interface is customizable and supports synset browsing and editing, hypernym/hyponym relations are visualized in a tree structure. VisDic also includes tools to merge wordnet parts edited by several users. After the Balkanet project, it was used by several national wordnet projects. VisDic is not actively developed, but still supported and used by wordnet authors. VisDic is a desktop application, working on Linux and Windows.

**DEBVisDic** A successor and a complete reimplementation of VisDic system [HS03], based on the DEB platform (see chapter 7.4 for more details). DEBVisDic is based on a client-server architecture, thus solving the main disadvantage of VisDic in team development of wordnets. DEBVisDic supports all functions of VisDic and adds new features, for example integration of Visual Browser [Nev07] to visualize semantic networks, and connection to external resources. It was already used to build over 20 national wordnets, for example a group of South African languages in a project by North-West University. DEBVisDic is not only the wordnet editor, but also a flexible tool for analysing and editing structured knowledge resources. Thanks to the flexibility, it was used in Cornetto project (described in section 7.5) and in international KYOTO project (described in section 7.6). DEBVisDic is based on a client-server architecture, with desktop client working in any operating system, and later transformed into a full web application.

**WordNetLoom** Wordnet editing tool was developed for the construction of Polish wordnet, called plWordNet, at Wroclaw University of Technology [PMRM13]. WordNetLoom provides two modes of operation, either a “form-based” editing of synsets, or “graph-based” method for manipulation with synset lexico-semantic relations. WordNetLoom is a desktop
application, independent of operating system, and the MySQL database is required for the data storage.

**HYDRA**  An application developed during the building of Bulgarian wordnet, called BulNet, at Institute for Bulgarian Language [Riz14]. Hydra tool supports functionality for wordnet creation and editing, similar to other described systems. As a distinctive feature, Hydra supports modal language for wordnet querying. Hydra is a desktop application, tested in Linux and Windows operating systems, with the data stored in MySQL database.

**WUpdate**  This is not a full-fledged wordnet editor, but rather a data manipulation language designed to perform wordnet modifications (e.g. adding senses, or linking synsets) [Kub14]. WUpdate is part of WQuery [Kub12], query language for wordnet-like databases. Together with the language model, a set of tools to search or update wordnets is included in the WQuery suite.

### 5.3  User experience and dictionary personalization

Many research projects were carried out in regard to dictionary usability and the impact of dictionary design, structure, or content on the different tasks required by different groups of users. For example, a difference in word comprehension by learners using either a learner’s dictionary or a native speaker dictionary. For various examples of dictionary use research, see the following papers [LK97, Chr97, Var02, LH97, HA98, PL14].

With the growing popularity and availability of electronic dictionaries, the focus of research is extending to comparison of usability and contribution to word comprehension between printed dictionaries and electronic dictionaries, e.g. in the research [Dzito, KT04]. Recently, the research area is even shifting to comparing several electronic dictionaries or different methods of dictionary access among themselves, for example research by [Cheio, Tonoo, HL03].

However, the important question of electronic lexicography is not whether the printed or electronic dictionaries are better, but how can modern electronic dictionaries help their users with the required tasks. Since the beginning of electronic dictionary spread, publishers were not limited by the space in printed dictionary and the electronic version of the dictionary is usually an extended version of the printed edition, for example using full labels instead of abbreviations.

Of course, the descriptive labels help unskilled users of dictionaries to better navigate the dictionary and find information in an entry structure. Apart from displaying full labels, there are even more ways the electronic dictionary may help users. As discussed for example by Almind in [Almos]:

29
Help texts are important for the non-professional user. Let synonyms, antonyms, collocations and similar parts of an article be labelled clearly not with symbols but with proper headings. Let each heading be defined in the dictionary itself but also by linking it to a definition in the user’s guide which of course is one of the not to be forgotten texts the user can find on the site.

On the other hand, extending the amount of information (expanding and explaining labels, introducing more examples from corpora, etc.) may be too overwhelming for the users. The situation is even more distinct with the users moving from desktop computers to tablets and mobile phones, where the limited space to display the dictionary information is present. Lew in [Lew11] proposes to use different terms for the available size of dictionary data and the amount of information that is possible to display to users:

The suggestion that dictionary space is unrestricted is actually largely correct, but only when space is understood as the capacity to hold the total content of the dictionary – this sense of dictionary space could provisionally be called storage space. There is at least one more important sense of dictionary space which I will here call presentation space...refers to how much can be presented (displayed, visualized) at a given time to the dictionary user.

Current research in lexicography field is aiming to design and develop tools to allow high level of customization by users, both in the terms of dictionary presentation, and data provided. Unfortunately, there is not just one setting for each user, but different preferred ways depending on the given task. This requirement is summed for example by Tarp in [Tarp12, p. 115] as "...in the real world, the individual needs of individual users may differ from each other and that the concrete data to meet these needs therefore may be slightly different from user to user and from user situation to user situation", or Atkins & Rundell in [AR08, p. 245] as "Users have their own specific needs and skills (and these may change according to the task they are engaged in), so it is important to allow them to decide which information-categories should be displayed by default...".

Adaptability of the data and presentation based on users’ preferences is the task for electronic dictionaries. The first step in achieving this goal is giving users the possibility to select the “level” (e.g. by age, or proficiency with language) of dictionary they want to work with. This may reflect different editions of printed dictionaries, although it has been found that the distinction of dictionary level as drafted by lexicographers may not be consistent with users’ notions. For example, De Schryver and Prinsloo performed a study with three Dutch dictionaries and concluded that "the intended age group itself prefers the dictionary definitions compiled for the immediately lower age group" [dSPu1].
Probably the best way to achieve the desired level of user customization is to employ a lexical database with well-defined structure containing properly completed lexicographic data and develop the system that will select the right information and present them in a suitable way according to the user’s current requirements. This approach is discussed for example by Muller-Spitzer and Mohrs in regards to the OWID project (“the lexicographic contents are structured granulary and strictly content-based. This technology allows to define user-adapted views of the lexicographic data”) [MSM08], or in enhanced form by Spohr when describing the model of what he calls “pluri-monofunctional lexicographic tool, in the sense that it is capable of deriving multiple monofunctional dictionaries” [Spo11, p. 105]. In general, the requirements and aims of electronic dictionaries are described by Tarp as “the gradual development of highly sophisticated tools that permit both individualized access to the data contained in a well-structured database...and the recreation of completely new data based upon the already existing data”[Tar12, p. 115].

Currently, several available electronic dictionaries allow their users to select the amount of information and its presentation. There are several modes of the customization. Most widespread solution is to let the users choose from several pre-defined user profiles, usually a sort of ”short” and ”long” version of a dictionary entry. This type is used for example in Macmillan dictionary [Run07], or Den Danske Ordbog [HKo5]. A similar approach is used by several electronic encyclopaedias, offering the ”simple” and ”in-depth” versions of an entry, e.g. in Swedish National Encyclopedia [AB15]. The same method is used in the New Encyclopedia of the Czech Language, based on the DEB platform (for more details see section 7.8). Several dictionaries extended this concept to selection of profiles for different situations. For example a Danish phraseological dictionary [fL13] offers the selection from the following categories: 1) I am reading, but have a reception problem, 2) I am writing and want to use a particular expression, 3) I am writing and am looking for an expression with a particular meaning, and 4) I want to know everything about an expression. Most flexible type of customization is offered by Oxford English Dictionary, where the users are able to configure which information they want to see (e.g. pronunciation, or etymology). [TJ10]

For better user experience, the dictionary system monitors and analyzes user actions and adapts the interface and the data accordingly. However, a similar approach is not yet widespread in electronic dictionaries. One possible method was proposed by de Schryver in [dSt0], he suggests to enrich the dictionary with a set of rules to change the level, for example in the form of:

If the user mainly searches uncommon words then set the user’s level to expert, together with appropriate, numerical threshold for ’mainly’ (e.g. in at least 75% of the cases) and ’uncommon’ (e.g. occurs at most 20 times in a given corpus).

Adaptive dictionary interface may help not only the users to better search for the data, but
may also be beneficial for language learning and support language acquisition. Such dictionary interface, named RoLo (Remind on Lookup), was proposed during research by Dang et al. [DCD+13]. Experimental dictionary interface highlights collocations or meanings that the users previously explored, thus making the search faster and supporting the acquisition of new words in foreign language.

The results from the studies in this research show that RoLo improves not only incidental vocabulary learning, but also the ease and speed of re-lookup. Compared to conventional dictionaries, RoLo provides more effective support for understanding unfamiliar words and comprehending L2 texts.

On the other hand, the dictionary interface proposed in RoLo project focuses solely on learner’s dictionary and is not applicable generally.

A good starting point for the design is the analysis of the log files that demonstrate user behaviour and usage patterns of the published electronic dictionary, e.g. showing the frequently searched words, or the number of searches in one session. Analysis by De Schryver & Joffe found that "...most visitors tend to look up frequent items on the one hand, and sexual/offensive items on the other..." [dSJ04] and suggests that dictionary revisions may be successfully based on the log files analysis.

However, more recent study by Verlinde and Binon [VB10] found that "...there are no frequent recurrent look-up strategies..." and that user behaviour is dependent on the dictionary. To develop a smart adaptive dictionary interface, we need much detailed and comprehensive data on the dictionary use, combined with the information regarding users (e.g. language proficiency). However, no such research study was performed yet. To give the best results it should be implemented in cooperation by theoretical lexicographers and computational linguists.

5.4 Effective access to dictionary data

Assessing the recent research in lexicography and published proposed dictionary access models [VP12, Lew12, Tar08, AR08, Dzi14, Bog03, Haroi, Ton11], we may identify four features that the electronic dictionary needs to take care of to help users with effective access to the data.

Headword identification The users are often searching for inflected forms of the word. Sometimes users are entering misspelled words, either as a typo, or because they are not sure how to spell the word correctly. This feature is heavily dependent on the language and availability of other tools (e.g. morphological analyzer).

Accessing multi-word units It is difficult to find the multi-word expression in printed dictionaries, because users are often not sure which part of the multi-word unit is the main
dictionary entry. The electronic dictionary may offer the multi-word units when searching for just the part of the expression. However, users may not identify the multi-word expression in the text and try to search for the whole part of sentence that they do not understand. Dictionary system have to cope with this search and detect the words and multi-word units in the text.

**Type-ahead search**  Both for variant forms of headword and multi-word units, type-ahead search suggestions may help users to find the right dictionary entry. For example, when they know the few first letters of the word and are not sure about the right spelling of the rest of the word, type-ahead search will suggest the correct entry.

**Entry navigation**  As discussed in section 5.3, users may be overwhelmed by the amount of information available in the dictionary. Electronic dictionary provides methods for faster identification of the relevant parts of the dictionary entry. Proposed methods include a menu of target items (e.g. word senses, collocations) with the sequential displaying of selected items, or initial display of short snippets instead of full descriptions.

### 5.5 Linked Data

The term Linked Data refers to a methodology for publishing and interlinking structured data online. This methodology was proposed by Berners-Lee in 2006 [BL06, BHBL09], who outlined four rules data are required to meet for easy sharing and interconnecting:

- objects are identified by an URI¹ (e.g. http://dbpedia.org/page/Brno),
- URI identifiers are HTTP links, where people or software tools can access the data,
- useful information are provided on given URI, using the appropriate standards (like RDF) (previously mentioned page contains links to the same information in multiple formats, RDF is provided at http://dbpedia.org/data/Brno.rdf),
- other objects are referenced using their URIs to get more information (e.g. link from the Brno.rdf to http://dbpedia.org/resource/South_Moravian_Region).

### 5.6 Chapter conclusion

This chapter identified the requirements for dictionary writing systems and semantic network editing software. Existing applications in the field were explored and compared, together with current trends and methods for innovative electronic dictionaries, both from user experience, and effective using of dictionary points of view.

From the comparison of existing systems and applications, it is clear that all the software aim on one type of resources only. For better support of the language analysis and advanced

¹Uniform resource identifier [BLFM05]

33
research techniques, more widely aimed system is needed. Lexicographers, resource authors
and managers need the adaptable and universal system that supports multiple resource types,
follows recent lexicographic research in areas of user experience and effective data access, and
allows efficient sharing and combining of resources and data.

The following chapter describes the DEB platform developed by the Natural Language Pro-
cessing Centre with the thesis author acting as the main developer of the system and related
applications. The DEB platform features are compared with the requirements described in
this chapter.
6 DEB platform

The Dictionary Editor and Browser (DEB) platform offers a development framework for any dictionary writing system application. The DEB platform is being developed at the Natural Language Processing Centre, Faculty of Informatics, Masaryk University since 2005. Main advantages of the DEB platform are client-server model, open-source and freely reusable code, modular architecture, and versatility. The design and the development of the platform is the work of the thesis author, co-authored parts (mostly client software) are referenced in respective sections later.

Following sections compare the DEB platform with existing dictionary writing systems, describe the technology used and implementation, reference the reception of the platform and related projects, and describe several prominent projects that use the DEB platform.

6.1 Features

Following sections describe the DEB platform features and compare them with the dictionary writing systems, using the same categories as in section 5.

6.1.1 Overview

The DEB platform uses client-server model. All the data are stored on the server and a considerable part of the functionality is also implemented on the server, while the client application
can be very lightweight. This solution also supports homogeneity of the data structure and presentation. The data storage is based on XML standard [BPSM00] which allows flexible data structure and reusability of existing standardized markups and tools. The DEB platform itself and all the libraries and tools used are licensed under open-sources licenses that permits free use. The DEB platform source code may be downloaded from the project website, and the license permits its reuse and modifications for free. The project is developed and tested on the Linux operating system, however both the DEB platform, and all the libraries needed are multi-platform, thus the platform may be deployed even on MacOS or Windows. The installation packages are provided for the Ubuntu Linux distribution, but the DEB platform server part was successfully deployed on Debian and Fedora distributions.

Previous information applies for the server part. The client software is usually a web application, thus the users may access the data from any operating system with just the web browser. Various client applications were tested and regularly used on Linux, Windows and MacOS operating systems. General conclusion is that the server part is usually installed on the Linux operating system and client software is multi-platform.

Feature comparison (for analysis, see table 5.1, for comparison, see table 6.1):

- operating system: server part Linux (with the multi-platform possibility), client part multi-platform
- price: free
- support: provided by the Natural Language Processing Centre
- DWS type: client-server

<table>
<thead>
<tr>
<th>DWS</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Windows</td>
<td>Win, Linux, MacOS</td>
<td>Win, MacOS</td>
<td>Windows</td>
<td>Win, Linux, MacOS</td>
</tr>
<tr>
<td>price</td>
<td>1000 €+</td>
<td>400–7000 €</td>
<td>150–1900 €</td>
<td>150 €+</td>
<td>free</td>
</tr>
<tr>
<td>support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>type</td>
<td>client-server</td>
<td>client-server, single-user</td>
<td>client-server, single-user</td>
<td>client-server</td>
<td>client-server</td>
</tr>
</tbody>
</table>

6.1.2 Dictionary management

The DEB platform includes the web interface for dictionary and user management [HR07b]. It is possible to define the data structure of the dictionary, using the XML Schema. Apart from the data structure, various XSLT templates that are used for output formatting may be assigned to each dictionary. It is also possible to specify cross-reference links between several dictionaries. Defined dictionaries may be grouped together (and reused multiple times) for different projects.
Apart from the web interface, API interface is also provided to integrate dictionary or user management functionality in third-party applications.

Feature comparison (for analysis, see table 5.2, for comparison, see table 6.2):

- dictionary structure: free data structure
- add new dictionary: unlimited
- structure definition: XML Schema, with custom extensions for more variability

Table 6.2: Dictionary management, comparison with DEB

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictionary structure</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
<td>any</td>
</tr>
<tr>
<td>add new dictionary</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>structure definition: DTD</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>- XML Schema</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>- edit in GUI</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

6.1.3 User management and access control

The user management is also set-up using the administration interface described in the previous section. User accounts are shared between all the services, thus users need just one account for all the dictionaries or projects they may use. Each dictionary application may specify the groups and granularity of the access control, ranging from straightforward determination whether a certain dictionary is readable by the user, distinction between several user roles (e.g. reader, editor, supervisor,...), to complex access rights system combining user roles with different user groups. For example, complex access rights may define several departments, each responsible for editing a different part of a dictionary entry, or a different dictionary domain, and users may get various roles (e.g. editor, reviewer, supervisor...), the DEB platform is then able to combine all the access restrictions.

Access control also enables different scenarios for data sharing across users or institutions. Using the API provided it is possible to share dictionaries stored in the DEB platform both with other DEB platform servers, and third-party applications. A project manager or administrator may decide that selected dictionaries will be shared publicly, while other dictionaries will be shared only with a restricted set of authorized users [HPR08].

User management and authentication may also be linked to the external service, thus allowing to reuse existing user accounts. For example, user accounts may be linked to an institution’s LDAP service [TGM04].

Feature comparison (for analysis, see table 5.3, for comparison, see table 6.3):

- user accounts: possible, one account for each user shared for several applications
• external accounts: possible
• user roles: possible, from straightforward distinction to complex system

**Table 6.3: User management, comparison with DEB**

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>user accounts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>external accounts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>user roles</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**6.1.4 Writing process management**

The DEB platform provides the entry locking mechanism for concurrent editing. Whenever a user starts editing an entry, that entry is accessible only for that user. After the edits, the user will save and release entry for all others. Locked entries are also released for editing when the user logs out from the application. In case of unexpected events (e.g. network connection failure), the server will release the entries after a specified amount of time (the setting may vary between applications). It is also possible to get a list of locked entries through the API or web interface.

For more complex needs, the DEB platform also supports the work assignment. For example, the user with specified role (e.g. supervisor) may assign dictionary entries to different editors. The editors will see the list of entries to work on in the application, and are not able to edit other entries (depending on the access setting, they may not even see other entries).

The DEB platform server keeps track of every change, with the possibility to revert to previous versions of a database entry. If needed, various statistics and progress reports may be generated from the database and presented in the application.

Feature comparison (for analysis, see table 5.4, for comparison, see table 6.4):

• entry locking: possible, each application may use a different setting
• assign work: possible, based on access control
• progress statistics: possible, defined by application
• editing history: complete history stored on server

**Table 6.4: Dictionary management, comparison with DEB**

<table>
<thead>
<tr>
<th></th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry locking</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>assign work</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>progress statistics</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>editing history</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
6.1.5 Database engine and searching

Because the DEB platform was designed with the aim to be free, open-source, and multi-platform, the database engine must meet this requirement, together with features required to integrate with the system. It was also decided that it is pointless to develop a new database engine from scratch, and rather use an existing one and concentrate on developing the dictionary platform.

At the time of the initial analysis, the only database engine meeting all the criteria was the Berkeley DB XML [DB 07]. However, few years later when the number of applications and stored data grown, it became apparent that the database searching is slower on large data collections, even though the database access was optimized. For that reason, the analysis and benchmark of current XML database engines was performed, details are described in section 6.2.3. As a result of the benchmarking, Sedna XML Database [FGK06] was selected as the new database engine for the DEB platform.

Using the API or client applications, the users may specify all kinds of database queries, ranging from simple predefined search queries to complex XPath queries. All the search queries are internally converted to XQuery and executed on the database engine.

Feature comparison (for analysis, see table 5.5, for comparison, see table 6.5):

- database: currently Sedna XML Database
- stored as XML: native XML database
- complex search: yes, converted internally to XQuery
- regular expressions: yes
- XPath, XQuery: yes, supported by the database

<table>
<thead>
<tr>
<th>database</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>stored as XML</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓“XML-based”</td>
<td>✓</td>
</tr>
<tr>
<td>complex search</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>regular expr.</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPath, XQuery</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 6.5: Database engine and searching, comparison with DEB

6.1.6 Output formats

The DEB platform stores all the dictionary data in the XML format and it is possible to get the entries in XML format either using API or the client applications. JSON format is also supported for the data exchange through API.
XSLT templates are used for output formatting, with the option to assign many different templates to each dictionary. These templates are used for all possible entry representations, e.g. editing in client application, online browsing, output for publishing. Updates to the templates are immediately available in all the client applications, helping with the homogeneity of the data presentation.

Several client applications introduced the options to customize the output template in the graphical interface, for example selecting different colours for parts of a dictionary entry, or switching output templates.

Thanks to the XSLT and XSL-FO standard, it is possible to provide data in various formats for publishing. So far, a direct export to \LaTeX{} and PDF, and an indirect export to specifically formatted HTML that was later imported into DOCX format were tested.

Feature comparison (for analysis, see table 5.6, for comparison, see table 6.6):

- XML export: yes
- XSLT: yes
- GUI setting: partially
- electronic format: yes
- publishing format: \LaTeX{}, PDF, import to DOCX

### Table 6.6: Output formats, comparison with DEB

<table>
<thead>
<tr>
<th>XML export</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSLT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GUI setting</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>electronic format</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>publishing format</td>
<td>PDF, InDesign</td>
<td>✓PDF</td>
<td>✓PDF, InDesign, Quark</td>
<td>✓RTF, DOCX, DSL</td>
<td>✓LaTeX, PDF, DOCX</td>
</tr>
</tbody>
</table>

#### 6.1.7 User interface

The user interface is highly dependent on the client application. Several different approaches were implemented in various projects, and the most popular way to edit the data is the "flat" entry editing form. With the technologies used, it is possible to let users customize the entry editing form based on their preferences. It is also possible to localize the client application interface to different languages.

Feature comparison (for analysis, see table 5.7, for comparison, see table 6.7):

- entry form: yes
- tree form: yes
• editable preview: yes
• customizable: yes
• localization: yes

Table 6.7: User interface, comparison with DEB

<table>
<thead>
<tr>
<th>Feature</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry form</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>tree form</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>editable preview</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>customizable</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>localization</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

6.1.8 Helper tools

The DEB platform supports many ways to make dictionary editing easier for the users. For example, it is possible to define quick templates and insert them quickly to the edited entry. The applications are also checking if the correct cross-references (or bibliography references) are included in the text. When the referenced entry is updated, all the cross-references are automatically updated, too.

All kinds of multimedia content are supported, various dictionaries usually include images and pronunciation sound recordings. Most noticeable use of multimedia are the video recordings in the dictionary of the Czech Sign Language (see section 7.3 for details).

Thanks to the modular design of the DEB platform, it is possible to extend both server and client parts with various third-party applications. For example, majka morphological analyzer [JKŠ11] is included for Czech lemmatization in dictionary search.

Feature comparison (for analysis, see table 5.8, for comparison, see table 6.8):

• defining dictionary: possible, not yet requested in any application
• quick templates: yes
• cross-references: yes
• multimedia: yes
• virtual keyboard: yes
• corpus integration: yes

6.1.9 Ontology and semantic network editing

Based on the experience from the development of the VisDic dictionary software, one of the initial goals for the DEB platform was to support and enhance the creation, editing, and presentation of ontologies and semantic networks. For this purpose, the DEBVisDic application
Table 6.8: Helper tools, comparison with DEB

<table>
<thead>
<tr>
<th>Feature</th>
<th>DPS</th>
<th>iLEX</th>
<th>TLex</th>
<th>ALC</th>
<th>DEB</th>
</tr>
</thead>
<tbody>
<tr>
<td>defining dict.</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>quick templates</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>cross-references</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>multimedia</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>virtual keyboard</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>corpus</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

was developed. The application started as a reimplementation of the VisDic tool, but it also offers more advanced features. Currently, the client application of DEBVisDic is a web application usable in any browser, thus making it multiplatform and the users are not required to install any software. DEBVisDic is described in detail in section 7.4.

DEBVisDic application is designed to work with any ontology or semantic network. It is mostly used to browse or edit various wordnets, but it is possible to alter the entry structure to suit different semantic networks. For example, SUMO ontology is available in DEBVisDic and interlinked with wordnets. Another example of significant DEBVisDic customization is the building of Cornetto, complex semantic network merged with a lexical database (see section 7.5 for more details). DEBVisDic server also provides API interface that may be used to integrate ontology editing in third-party applications. Prominent example of such integration is the KYOTO project (described in section 7.6).

6.1.10 User experience and dictionary personalization

The DEB platform provides various tools to enhance user experience, that are utilized by different applications to fit the needs of given project. Generally, users may customize which dictionary information and in what form they want to see while browsing. For example, in the Czech Sign Language dictionary (for more details, see section 7.3) the interface allows the users to choose which parts of a dictionary entry to hide or show, or to select the information acting as a main entry representation from several options.

It is also possible to change the content and layout of editing forms, based on user preferences, or user roles (used for example in Cornetto or Czech Sign Language dictionary).

A different approach is used in the New Encyclopaedia of the Czech Language. Readers may choose their proficiency level and according to the setting, they will see different entry descriptions, or even different entries. The Encyclopedia is not published yet. After the public release, usage data will be gathered and analyzed. Behaviour patterns will be detected for users with different proficiency levels, for example what entries they search for, or whether they get what they were looking for. Based on the analysis results, the options will be evaluated to implement adaptive dictionary interface to provide even better experience for the readers.
6.1.11 Effective access to dictionary data

Section 5.4 identifies the main features that enable users to access the dictionary data effectively. A description of the DEB platform solutions for the requirements follows.

**Headword identification**  Thanks to the modular architecture and possibility to integrate third-party tools, the DEB platform may use existing morphological analyzers to identify inflected forms or misspelled words. Currently, majka morphological analyzer is used to enhance dictionary search in the Czech language for inflected forms of dictionary headwords. Very specific example that has to deal with correct headword identification is the dictionary of Czech Sign Language. Since the main form of the headword is a video recording showing the sign, it may be difficult for users to search for the dictionary entry. Each video recording is described by formal notation, that may be used for searching. Also simplified graphical search is provided for the users who are not familiar with the formal notation. The solution is described in more detail in section 7.3.

**Accessing multi-word units**  The DEB platform allows users to search for a multi-word expression just by entering a part of an expression, and the multi-word units are displayed both if they are separate entries, and subentries of the main headword. For more flexibility, users may choose to search in any part of a dictionary entry, e.g. in definitions or usage examples. Based on the analysis of the log of searches in the dictionaries provided by the DEB platform, users tend to enter a single word (which they think of as a main part of the expression) as a search when they search for multi-word expressions.

**Type-ahead search**  Search suggestions are provided by the DEB platform, based on the dictionary setting. Type-ahead search suggestions are provided not just for the readers, but also for dictionary editors, for example when entering cross-reference links.

**Entry navigation**  This feature is closely related to the previously described approaches to dictionary personalization by the users. If the users like to see less information, they may select different proficiency level. For complex entries (for example in lexical databases), basic information is provided by default and users may display more detailed information for various parts of the dictionary entry.

6.1.12 Quality control

The dictionary writing systems usually provide various methods for quality control of the dictionary data. However, the dictionary writing systems that are working with general dictionary
data, usually check only the cross-references between entries and do not cover special require-
ments for different dictionaries. With the modular architecture of the DEB platform, it is pos-
sible to define customized quality validations for each dictionary. The validations may be used
during the dictionary creation for immediate checks, or in pre-publishing phase to assure that
the quality of data given to the publisher meets the required standards.

In the Family Names in Britain and Ireland project (see chapter 7.1 for more details), the
quality control is implemented both in the editing, and pre-publishing phase. During the
dictionary editing, cross-references between entries, bibliography references, and correct en-
try formatting is checked. More specific data validations are included, for example there may
be variant spellings for each family name and the application checks that at least one of the
variant spellings contains the explanation. Another set of validations is run before sending the
data to the publisher (Oxford University Press). For example, the correct publishing layout or
completeness of the whole dictionary data is checked in this phase.

For the New Encyclopaedia of the Czech Language (described in detail in chapter 7.8), the
documents provided by the entry authors are validated during the import phase and later in
editing phase. Validations include checking the right format of entries prepared offline by the
authors, correct references to entries and the bibliography, or the presence of all the multimedia
files.

The dictionary of Czech Sign Language (see section 7.3) is produced by several teams, each
covering different parts of a dictionary entry. Thus validations are needed to check the com-
pleteness and correctness of entered information for each dictionary entry. The first part of
the validation is provided during the editing, for example correct references between the sign
language and the Czech language. More validations are run after the batch of entries is handed
over by a group of authors to the reviewers. During the review phase, reviewers may combine
several conditions to form complex validation queries. For example, the reviewer may check for
entries that contain video recording of the lemma and the definition, but not contain the usage
example or translations. A possibility to run complex validations after each phase is essential
for the successful completion of the dictionary.

These are just a few examples of the possible customization of the quality control methods
for the dictionaries produced with the DEB platform. The system provides many more options
and may be updated for various dictionary projects.

6.1.13  LINKED DATA

It is possible to publish resources stored in the DEB platform using the Linked Data method-
ology (as defined in Chapter 5.5). The DEB platform provides the tools and the decision how
to release the data lies with the author. Linked Data requirements are satisfied in the following
manner:
• use URIs as names – each entry has a unique URI identifier,
• use HTTP URIs – through the DEB platform API, entries are accessible on HTTP URI,
• provide useful information using standards – when linking to an entry URI, the data are displayed either in raw XML format, or converted to RDF or other defined format,
• link to other URIs – the DEB platform enables the links to other resources if provided by the data author.

These requirements were fully embraced in the DEBVisDic and KYOTO project (see chapter 7.6), where all the information were released as Linked Data. See Listing 6.1 for the example of information conforming to Linked Data methodology.

Listing 6.1: Information for synset forest, woodland, timberland, timber in RDF N-Triples format (shortened).

```
```

6.2 Technology

The DEB platform follows the client-server architecture design (see Figure 6.1 for the schema), which provides the ability of distributed authoring teams to work fluently on one common data source. The actual development of applications within the DEB platform can be divided into the server part (the server side functionality) and the client part (graphical interfaces with only a lightweight functionality). The server part is built from small parts, called servlets, which allow a modular composition of all services.

Client applications communicate with servlets using HTTP requests in a manner similar to recently popular concept in web development called AJAX (Asynchronous JavaScript and
XML [RM98]) or using the SOAP protocol [ML07]. The data are transported over HTTP in a variety of formats – RDF, XML documents, JSON-encoded data [Cro06], plain-text formats, or marshalled using SOAP.

6.2.1 Server software

The server side of the DEB platform is implemented in the Ruby programming language, which is an object-oriented, interpreted, general-purpose programming language with weak type checking [FM08]. The DEB platform is implemented in Ruby version 1.8, with the plan to refactor the modules in near future to use currently available version 2.1 of Ruby. The main advantage of the newly available version for the DEB platform is a much better support for Unicode in the Ruby core libraries. Currently, third-party libraries are used to work with Unicode strings and alphabetical sorting according to various national standards (TwitterCldr).

The database storage engine must meet the same criteria as the DEB platform itself – free, open-source, and multi-platform. Other feature requirements for the database engine included easy integration, support for XML-related standards, and of course the effective storage and searching in large collections of XML documents.

---

1 Implementation of the ICU standard (International Components for Unicode), https://github.com/twitter/twitter-cldr-rb
At the time of the initial analysis, the only database engine meeting all the criteria was the Berkeley DB XML [DB 07] (later acquired by Oracle and renamed to Oracle Berkeley DB XML). However, few years later when the number of applications and stored data grown, it became obvious that the database searching is slower on large data collections, even though the database access was optimized. For that reason, the analysis and benchmark of current XML database engines were performed, details of which are described in section 6.2.3. As a result of the benchmarking, Sedna XML Database [FGK06] was selected as the new database engine for the DEB platform. Thanks to the modular design of the platform, the change of the database was done by implementing a new database connection module, without the need to rewrite other modules or applications. The database interface for the Ruby programming language is provided, so it is possible to manipulate with the documents directly within the DEB platform modules.

6.2.2 Client software

The user interface that forms the most important part of a client application usually consists of a set of flexible complex forms that dynamically cooperate with the server parts. It is possible to implement client applications in any programming language that allows to interact with the DEB server using the available server interfaces. Client applications developed as part of the DEB platform utilizes two main approaches.

Initially, the DEB platform has adopted the concepts of the Mozilla Development Platform [Fel07] for client software. The Mozilla platform provides a complete set of tools for GUI software development. Firefox Web browser is one of the many applications created using this platform. The Mozilla Cross Platform Engine provides a clear separation between application logic and definition, presentation and language-specific texts. The application design is simple and allows the possibility of concurrent work of different team members which leads to saving time. Important reasons to select the Mozilla platform were the possibility of multi-platform development and the rich scale of available GUI features. Client applications were implemented as extensions installable in the Firefox (or other Mozilla-based) browser. Since the Firefox browser was already widespread at the time, most users did not need to install specific software for the DEB platform client applications.

The main “programming language” used for the GUI design of the DEB platform clients in the Mozilla Cross Platform Engine is called XUL (XML User Interface Language [HGW01]). XUL is a user interface description language based on XML. It allows rapid creation of cross platform applications with the possibility of easy customization of design, texts, and localization. XUL itself is aimed only at the creation of user interface, e.g. windows, buttons, or toolbars, and incorporates wide range of standardized technologies.
Cascading Style Sheets (CSS) [Eten1] for describing the graphical appearance of the application,
JavaScript [ECM11] as a programming language for simple application logic,
Document Object Model (DOM), XSLT and XPath to work with HTML and XML documents,
DTD for easy localization,
RDF as data source.

All of these technologies allow to build client GUI applications that fit to the DEB platform approach and are successfully deployed in many knowledge resource projects. However, there is a small portion of users that are not able to use Firefox browser, usually due to security policies at their institution. Over the years, more web browsers gained larger share of users, so the implementation of the DEB platform client software as a Firefox extension is not so generally available as before. On the other hand, the development of web-related technologies and their implementation in various web browsers is moving forward very fast. Thanks to the new standardized technologies available, it is now possible to implement complex GUI client software as the web applications that may run in any modern web browser. Also, the earlier client software is now re-implemented from the Mozilla Cross Platform Engine to the rich web application (one of the examples is re-implementation of the DEBVisDic application, described in section 7.4), making it available for even more users in easier way.

Rich web applications are implemented using the standardized HTML and CSS technologies for data presentation. JavaScript is used for the application logic and communication with the DEB server.

Because the implementation of web-related standards (mainly JavaScript) may vary slightly in different browsers, several toolkits and libraries provide a unified environment on top of the browser interface. Various DEB client applications tested several libraries. Currently, the two approaches are used for the client software development. The first approach for applications that need better control of the user interface or require specific features is to use either core JavaScript features, or the jQuery library [jQu15a]. jQuery is the versatile and lightweight JavaScript library for simpler document and data manipulation, and AJAX communication. The second approach for applications that need complex forms for editing dictionary data utilizes the ExtJS library [Sen15]. ExtJS is the JavaScript framework providing complete range of elements to build GUI applications in the web browser. It also offers various methods for server interface connection and data manipulation.

6.2.3 Evaluation of XML-based databases

The results of the evaluation of XML-based databases was published as a paper [BHR10] in the proceedings of the KES 2010 conference.
Database systems working with XML data (both native XML databases and XML enabled relational databases) are already widespread and used in many areas. Their performance was benchmarked by many projects using several benchmarks. In [BRo8], a generally applicable benchmark XMach-1 is described and compared to other benchmarks. Results for several databases are presented, showing that native XML databases perform better than XML-enabled relational databases. Unfortunately, no database is named, so the results are only general.

Nambar et al. [NLB+02] use XOO7 benchmark to compare several XML-enabled and native XML databases. Their results suggest that XML enabled relational databases process data manipulation queries more efficiently. Native XML databases, on the other hand, are more efficient in navigational queries which rely on the document structure.

Extensive comparison experiments were conducted by Lu et al. [LYW+05]. Their results suggest that different XML benchmarks can show different weak and strong points of each database system.

Differences in the results lead to the conclusion that customized XML benchmarks are needed in addition to a general XML benchmark to fully test the requirements of the application developed. For example for the business XML systems, Nicola, Kogan and Schiefer in [NKS07] offer specialized benchmark, called “Transaction Processing over XML” (TPoX). This benchmark aims to provide good comparison of XML databases suitable for the business process modelling.

Although there are many native XML databases, the selection was limited to databases that correspond to the licence and technologies applied in the DEB platform. The most important features are the open source licence, active development and support of XML-related standards.

From more than 20 native XML or XML-enabled databases, the following four systems were selected according to the designated requirements.

**eXist**

The eXist database [M+03] is developed in Java and licensed under LGPL, active since 2000 and currently developed by the group of independent developers. The database supports XQuery, XSLT and XUpdate standards for data manipulation, and DTD, XML Schema, RelaxNG and Schematron for validation.

Users are able to specify structural indexes (element and attribute structure in documents), range indexes (contains, starts-with and similar functions), and full-text indexes (Apache Lucene engine [Fou06] is used for full-text indexing).
**MonetDB/XQuery**

The MonetDB/XQuery database [BGvK+06] is developed by CWI Amsterdam and several Linux distributions and MS Windows are officially supported. The database is licensed under a customised Mozilla Public License.

The main goal of MonetDB is to design a database for processing very large (in GBs) XML documents. The default database settings are optimized for document reading, and offer indexing for quick query execution, although the indexes have to be rebuilt after every document update. Another option is an optimization for document updating with simpler index structure and slower performance for search queries.

The database supports XQuery and partly XQuery Update [W3C09]. It is also possible to use MonetDB internal query language. Indexing is automated, without the possibility to alter settings in any way. The PF/Tijah [HRvOF06] text search system is utilized for full-text searching.

**Sedna**

The Sedna database system [FGK06] is developed by the Russian Academy of Sciences, and released under Apache Licence. Official packages for Windows, Linux, MacOS, FreeBSD and Solaris are available.

The database supports XQuery and custom variant of XQuery Update for data manipulation, and XML Schema for validation. Indexes have to be set manually and a special function must be used in the query to access the index. Full-text indexing is provided by external commercial tool dtSearch. Sedna offers several extensions, such as the capability of an SQL connection from XQuery, or the trigger support.

**Oracle Berkeley DB XML**

Berkeley DB XML [CRZ03] was created as an extension of Berkeley DB. Currently, the database is being developed by Oracle and released for Windows and Linux. Users can choose between open source and commercial licence.

The underlying structure is still based on Berkeley DB and each document container is stored in a single file. The database supports XQuery and part of XQuery Update. The document validation according to a supplied XML Schema is checked only during a document storage, later changes can render the document invalid. Users have to specify indexes manually, full-text indexing is also supported, although it is not possible to use regular expressions in queries.
Because of the special focus on dictionary writing systems, two different test suites were utilized. For the general database performance, the XMark benchmark \cite{CWI09} was selected, and for the knowledge and ontology test, a custom set of the most frequent queries and tasks was prepared.

Following database versions were used in the evaluation (preferring stable release over the development one): eXist 1.4.0, MonetDB/XQuery 2009-Aug-SP1, Sedna 3.2.91, and Oracle Berkeley DB XML 2.5.13.

The XMark Tests

The XMark benchmark was developed in CWI Amsterdam with the aim to provide a benchmark suite for users and developers to choose the right XML database and to tune the database settings.

The benchmark includes the tool xmlgen to generate an XML document of a given size. The data and the structure are always the same, users are able to change just the document size. The test suite itself consists of 20 XQuery queries that model different operations with several collections of XML documents, ranging from simple search to complex linking and result generation.

The queries were tested on documents of size from 1.8MB to 114MB. You can see the results for the smallest and the largest document in Figures 6.2 and 6.3. For the smallest document, all the queries were executed in less than a second, except for several queries in eXist and Berkeley DB XML. The problematic queries q11 and q12 are combining data from two collections and
building very large result set. Although this is a complex task, it should not take so long on such a small document.

On the other hand, it is understandable in the case of the large document. With increasing the document size, the execution times are getting longer. For the 114MB document, much more queries are carried out in times above one second. MonetDB is providing the best results for large documents, and Sedna can be better for less complex queries on smaller documents.

Although the results of XMark tests can help users pick the right database, real data and tasks should be taken into account, because the results vary significantly according to the document size and query complexity.

Knowledge and Ontology Data Benchmark

Another step was to test the databases on real data and most frequent tasks of DEB applications. For the benchmark, the following lexicons and ontologies were used:

- The Dictionary of Literary Czech (SSJC), 180,000 entries,
- The Reference Dictionary of Literary Czech (PSJC), 200,000 entries,
- The Dictionary of Written Czech (SSC), 49,000 entries,
- The Dictionary of Words with Foreign Origin (SCS), 46,000 entries,
- The Dictionary of Czech Synonyms (SYNO), 23,000 entries,
- The Dictionary of Czech Phraseologisms and Idioms (SCFIS), 14,000 entries,
- The English WordNet (WNEN), 117,000 entries,
- The Czech WordNet (WNCZ), 28,000 entries.
Figure 6.4: Average time (in seconds) for the equality query

The operations were analyzed and the most frequent query types were selected, as well as several queries requested by the users.

**Equality Query**

In the first run, XQuery was used to select entries with an element equal to a given value. In the second run, the query was rewritten as an XPath query. With this optimization, databases performed much better, significant improvement was seen for eXist and Berkeley DB XML. The results are shown in Figure 6.4.

**Full-text Search**

A more or less standard data set for full-text benchmarks is the INEX collection [SSK07]. The current version of INEX collection 2009 contains 2,666,190 semantically annotated Wikipedia articles. The full-text search over the INEX database tests the database performance with a huge amount of data and complex-linked full-text structure. The tested databases often have problems with the kind of data structures used in INEX (e.g. Sedna was not able to build indexes for INEX at all, eXist did not return answers to many queries, MonetDB could not load the databases into 4 GB of memory). However, for the purpose of dictionary applications, the full-text search is usually used within short texts, such as definitions or examples. Thus the comparison of full-text search over standard dictionary tags is more relevant for the database evaluation.

For the eXist database, the Lucene module was used for full-text search. It was not possible to install PF/Tijah module on the testing server for MonetDB. And for Sedna, the commercial
module was not tested. The results are shown in Figure 6.5.

Considering that full-text modules for MonetDB and Sedna were not used, it is surprising that these databases processed the queries in times comparable to eXist (sometimes even faster). Berkeley DB XML results are missing for most of the dictionaries, because several queries of the test suite were not completed in five minutes.

Document Updates

Because the DEB platform applications are designed for editing the knowledge and ontology data, the documents are updated by teams of users. Another feature to test is the performance during document deleting and saving. For this test, only the largest dictionary (PSJC) was used. The test was run several times and each time five random documents were deleted and then saved again.

The average results are shown in Figure 6.6. Surprisingly, the differences between the databases are quite significant.

Evaluation

According to the results of the tests, none of the available native XML databases can supersede the others for all kinds of operations needed for knowledge and ontology storage and manipulation. Berkeley DB XML cannot efficiently solve the queries involving multiple nodes and full-text queries. The eXist database contains the Lucene module for text search and supports many XML standards, so it can be recommended for deployment where these features are more
important than the database performance. On the other hand the MonetDB database can be, according to its specific architecture, conveniently used when working with very large amounts of XML data. For middle-size data collections, the Sedna database can provide the same performance as MonetDB, while offering a richer set of features. The potential drawbacks of Sedna are the need to use special queries for the defined data indexes and the use of a commercial tool for optimized full-text queries. However, the full-text queries without this optimization are already comparably fast.

Evaluation results

Following the benchmarks, the integration of Sedna and MonetDB databases with the DEB platform was tested. Since the Sedna database was much easier to install and integrate with the rest of the DEB platform modules, it was selected to be used as the main database storage for the DEB platform. Sedna XML database is used since 2010 and the performance analysis with very large datasets, integration features, and the support for knowledge resource data proved that this was a reasonable decision.

6.3 Reception and use of the DEB platform

The DEB platform as a whole and its various applications and projects received a lot of feedback and mentions. Recently, the DEB platform is described in the dictionary writing systems overview by Abel in [Abe12] as a system that "deserve particular mention" because of the range of different ap-
applications, i.e. covering more than dictionaries, but also semantic networks, or encyclopaedias.

DEBVisDic is used as a tool to build and expand the group of wordnets for the South African languages (four Bantu languages, plus Afrikaans), currently in the third phase of development and adding the 5th African language \[PBF08, GB14\]. It is also used to create Polish wordnet PolNet, currently in the second version \[VK14\], the Filipino wordnet \[BPRD10\], the Open Dutch WordNet \[PV14\], the Czech wordnet, the Nepali wordnet, the Ukrainian wordnet, the Hungarian wordnet, or the Brazilian Portuguese wordnet OpenWordNet-PT \[dPRdM12\]. The DEBVisDic application was customized to build the Cornetto lexical database, described in section 7.5, or to provide backend ontology and knowledge database storage for the KYOTO project, described in section 7.6.

Even if DEBVisDic is not used to build a wordnet and the specific tool is developed for various reasons, authors compare their tools with the DEBVisDic, explaining the decision to use another tool. However, DEBVisDic is still a software used to create wide range of wordnets, and other applications are developed specifically to create one national wordnet.

DEBVisDic was used to build the Slovenian wordnet sloWNet \[FN11\], however due to the need to install the client software in the first version of DEBVisDic, the team developed their own tool.

The authors of the Polish wordnet plWordNet mention that DEBVisDic was not available when they started the project, so they had to build their own tool, later comparing the tool with DEBVisDic – "...it was not as universal and flexible as DEBVisDic, but it implemented the assumed procedure of linguistic work." \[PMM+10\]

The team creating the Bulgarian wordnet developed their own tool (Hydra) that meets their requirements, but mentions that "VisDic and lately DEBVisDic are very good and famous tools for wordnet development..." \[Riz\]. Consequently, Hydra supports DEBVisDic XML format for data import and export.

For the Ancient Greek wordnet, currently in the first phase of building, several applications (including DEBVisDic) were considered, however authors "decided that some peculiarities of the targeted language need to be managed accurately" and that it would be properly treated by specifically developed tool \[BBDG+14\].

Various wordnet projects, for example Slovenian wordnet sloWNet \[FN11\], automatically translated French wordnet WoNeF \[PdCBDt14\], or French wordnet WOLF \[SF+o8\], or software tools, for example Hydra \[Riz14\] or WordnetLoom \[PMRM13\], provides their data in the DEBVisDic XML format, making this format a widely used standard for wordnet exchange and sharing.

The author of WQuery querying language \[Kub12\] (described previously in chapter 3.5) compared his proposal with the DEBVisDic querying features (both provided by API, and in GUI application). In the evaluation, he states it is not possible to search for two kinds of

56
queries with the DEBVisDic – “reachability query”, i.e. finding path that connects two synsets, and “least common subsumers”, i.e. the closest common hypernym of two synsets. While it is true that DEBVisDic application does not offer option to perform such queries, it is possible to perform the queries utilizing internal methods together with the XQuery tools on the wordnet stored in XML data structure. Next version of the DEBVisDic API and application, currently in development, will be extended for these types of wordnet queries.

6.4 Chapter conclusion

This chapter described the design and implementation of the DEB platform for the development of applications for language resource creation and management. Thanks to the modular design, the platform is not limited to one type of language or knowledge resources, and supports requirements for many frequently used resource types, while existing applications specialize only on one type of data.

The DEB platform also enables the use of features proposed by recent research in the lexicography and resource interchange, thus providing technology not widely supported by the specialized or commercial applications. The platform is designed and developed by the thesis author. As reviewed in chapter 6.3, the DEB platform and its applications are widely used in international research projects and set the standards and methods for resource sharing.

Next chapter demonstrates the variability and adaptability of the DEB platform by presenting selected projects that are based on the platform.
Selected projects

Following sections describe selected projects based on the DEB platform. Projects and applications were chosen to show various interesting features of the DEB platform. The design and implementation of the presented projects is the work of the thesis author, co-authored parts (mostly client software) are referenced in respective sections.

7.1 Family names dictionaries – combining big data resources

This chapter will describe two different projects that share the similar goal and methodology, and reuse the same tools. The earlier project is the Family Names in Britain and Ireland (FaNBI) \[HCM11\] by the University of the West of England that started in 2010, the first phase finished successfully in May 2014 and the research was extended to 2016. The other project is the preparation of the second edition of the Dictionary of American Family Names (DAFN2), lead by the chief editor Patrick Hanks and to be published by the Oxford University Press. The project started in 2014 and aims to finish in 2016.

FaNBI is a research project that aims to complete a detailed investigation of the origins, history, and geographical distribution of the 45,000 most frequent surnames in the United Kingdom. The DEB platform was selected by the University of the West of England (UWE) as the dictionary writing system for the project because of its versatility and possibility to combine various resources.
Editing and management application

A custom web application was developed for the editing of the dictionary content and management of the work progress. Apart from the research team at UWE, many consultants from various institutions all over the world are involved in the project. For this reason, the editing client software was designed as a web application for multi-platform use (see Figure 7.1). The application is implemented in standard HTML and JavaScript for the best compatibility with all modern web-browsers.

Each family name may have several variant spellings or derived names, and to keep all the information in one place, the names are grouped into "clusters" of related names. When the author wants to edit a family name, a complete cluster is opened in the editing application. It is possible to quickly move information (e.g. explanation) from one entry to another, add another name to the cluster, or select the main name for the cluster. The application will update cross-references automatically in such cases.

Each entry contains statistical information about the family name – the frequency of the name according to several records (data from 1881, 1997, and 2013 are included for Great Britain, and from 1997 and 2008 for Ireland) and the most prominent location where the name is present. Also the tools to keep track of the work progress and sharing comments or requests with other team members are provided. One entry may be composed of several senses describing the name origin, usually specified by the language or culture of origin (e.g. Welsh, Jewish, Arabic...). Each sense may contain several explanations describing the name origin from various point of views, several evidence of "early bearers" from historical sources (when possible, at least one person for each century and place where the name was frequent enough), and the
references and links to more resources. When appropriate the application provides templates for frequently repeated parts of the description. When explaining the family names origin, authors need to enter characters from various alphabets, a virtual keyboard is provided to select characters quickly. A convenient lookup for the family name in several resources, both stored in the FaNBI DEB database, or external webservice, is also available.

The application also provides tools for the project managers. Statistics of the work done by the authors are generated weekly. Users may also get a progress report for the whole project, either for the whole database, or a selected range. The report shows the current status of entries (e.g. how many of them are finished, not yet edited, etc.) and the number of entries in various categories (e.g. main entry, variant entry). Any comments or requests blocking the entries are reported, too.

**List of names and frequencies**

The list of frequency for each family name is the cornerstone of the FaNBI project. It is not only the list of entries to edit, but the frequency also decides which names will be edited in each phase of the project. In the first phase, all names with more than 100 bearers were edited. The work was extended to all names with more than 20 bearers in the second phase.

At the beginning of the project, two lists were used – 1881 census report [Arc] and 1997 statistical data [HC12]. However, both lists had to be preprocessed and filtered, because they contained a lot of noise and errors (for example, spelling errors or invalid characters). Another issue with the lists provided was that all the names were written in uppercase. A straightforward solution is to leave the first letter of each word uppercase and the rest in lowercase, however, this is not true for all names. For example, Scottish and Irish names like O’Brien or McGaffin had to be considered. This type of names also produced the issue with various written spellings used. For the Mc- names, three different variant spellings were present – Mac-, Mc- and M’. Similarly for O’- names, various apostrophe characters were used and sometimes the name was written without the apostrophe. It was decided to include only the spellings Mc- and O’- into the dictionary, and redirect readers searching for other variants to the correct dictionary entry. To make the matter even more complicated, some family names starting with the string Mac- are separate names and not the variant spellings of Mc-, for example Mach or Mackarel. To solve this issue, the list was edited in two steps. In the first step, variant spellings were detected and uncertain samples were reported. In the next step, the proposed changes were approved by the lexicographers. In case of variant spellings, the frequencies had to be summed for all the forms.

The 1881 census list was edited with the described method, and the method was updated for other lists. Lexicographers’ approval was not needed anymore, because the 1881 list was included in the cleaning tool to decide the correct spellings and variant combination. Finally,
only names with frequency of at least 20 were included. During the cleaning and combining of the 1881 census list, the number of records was reduced from 469,356 to 373,319 records.

With the report from recent years, the issue linked with growing immigration was discovered – both masculine and feminine forms of the names appeared for languages where these forms differ (for example, Polish or Czech). It was decided to keep only the masculine form of the family name, and the method for frequency inclusion was updated. Feminine forms are detected by the known suffix and if the masculine form is present in the database, the frequency numbers are combined.

Combining resources

In the aim to include as much historical evidence as possible, various existing databases are used to search for the records of the family names. A selection of records is available as a webservice from 'The National Archive', however it was needed to clean or preprocess the resources.

Very valuable resource for family names studies is the *International Genealogical Index* (IGI) [oCoLDS] compiled by The Church of Jesus Christ of Latter-day Saints. The IGI contains worldwide records extracted from the parish archives and similar sources, or submitted by the members of the Church. IGI records are published on the FamilySearch website, however the website does not provided access to the complete collection and records may contain various errors or inconsistencies. The original database records for the Great Britain were provided to the FaNBI project. The database was transcribed from the parish archives by volunteers over the course of several decades. Because of many reasons (for example, unreadable books, different spellings by each transcriber, spelling mistakes etc.) the database had to be cleaned up before it could be included in the FaNBI research. Sometimes, several volunteers transcribed the same parish records, so the duplicate data had to be detected. The following list sums the process of the cleaning and deduplicating the IGI database.

- Original database contained 188,043,185 records. Each record contains information about the event type (birth, christening, marriage, or death), first name, surname, date, location (county, town/place name, sometimes the exact parish), and the role of the person (e.g. for marriage bride, groom, or their parents).
- Obvious mistakes were deleted, for example records claiming that the English cities are in France.
- Names of the counties were standardized from variant spellings and abbreviations.
- For each county, a list of place names was extracted. These lists were distributed amongst the volunteers from the Guild of One-Name Studies. Volunteers checked if the place

---

2 [https://familysearch.org/](https://familysearch.org/)
3 [http://one-name.org/](http://one-name.org/)
name on the list belongs to the given county, or provided correct spelling. As a result of this process, a standardized list of place names was created and the database records were fixed. The records that provided incorrect information about the place name were deleted.

- In the next step, duplicate records were deleted. Because the main aim for the FaNBI research was not to build complete and perfect database, but provide reliable evidence, it was possible to delete not just exact duplicates, but also suspect duplicates. The rules for duplicate detection were considering following information from the records: first name, surname, date, town, county, and event type. Records were flagged as duplicate when all information were identical, but one of the following fields was different: first name, town, county, or event type.

- At the end of the process, IGI database contained 72,187,630 records.

Subsequently, the database was used to automatically add historical evidence to the FaNBI dictionary. For each family name, IGI records were extracted for each century and most prominent county, formatted according to the reference templates and saved in the entry. 40,274 family names entries were automatically enhanced with the IGI evidence. Apart from the enhancement of the dictionary, the processed IGI database is regularly consulted by the researchers as a valuable resource. For the sample of original IGI record and converted form to include as the historical evidence see Table 7.1.

**Table 7.1:** Original record from the IGI database and form included into FaNBI.

<table>
<thead>
<tr>
<th>Original record</th>
<th>Converted record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch identification, event date, event place, event type, year, first name, surname, role, gender:</td>
<td>John &lt;i&gt;Darter&lt;/i&gt;, 1629 in &lt;src&gt;IGI&lt;/src&gt; (Bletsoe, Beds)</td>
</tr>
<tr>
<td>Bletsoe, Bedford, England</td>
<td>05 Sep 1629</td>
</tr>
</tbody>
</table>

Another archive resource that required preprocessing were three volumes of *The Irish Fiants of the Tudor sovereigns during the reigns of Henry VIII, Edward VI, Philip & Mary, and Elizabeth I* [Nic94]. The Fiants contain various court warrants and are available in the electronic format. Each record is clearly marked in the text and thanks to the official language, it is possible to detect persons’ names, occupations, or residence. In the first step, the Word documents (results of the OCR recognition) were converted to the XML format. Each court record was converted into a separate XML entry with enhanced metadata. For example, the date of the record was converted from the regnal years system to calendar years.

Converted XML documents were later processed by the extraction tool. The tool standardized common OCR misspellings and detected frequently repeating text patterns in the warrant
texts. The list of place names (created during the IGI database cleanup) was used to detect town names and match them with the correct county. Where available, also the persons’ occupations were tagged in the record. Finally, all the information were formatted according to the FaNBI reference templates and are available for reference in appropriate entries. For the sample of the conversion from Fiants to FaNBI, see Table 7.2.

Table 7.2: Original Fiants record and form converted to include in FaNBI.

Original record:
1431. Pardon to Thomas Dowdall, of Dermondston, county Dublin, husbandman.— 2 November, xi.

Converted record:
Thomas <i>Dowdall</i>, 1569 in <src>Fiants Eliz</src> $1431 (Dermondston, co. Dublin)

Preparing data for the publisher

The resulting dictionary of family names will be published by the Oxford University Press (OUP), probably in 2016. During the development of the tools, the XML document format for the publication was discussed with OUP. Thanks to the design of the DEB platform output formatting, it was possible to test several prototypes before agreeing on the final delivery format. In case of updates (for example, enhanced dictionary data, or fixed spelling mistakes), it is possible to quickly prepare the updated data in the right format for publication. The dictionary data are validated before sending to the publisher. For example, the correct publishing layout or completeness of the whole dictionary data is checked in this phase.

From FaNBI to DAFN2

Since the DAFN2 project is quite similar to the FaNBI project and several authors work on both projects, the proven methodology and tools from the FaNBI project were adapted for the DAFN2 dictionary. The editing and management application is customized to meet slightly different requirements. Methods and tools developed for tidying the names frequency lists were reused, because the DAFN2 project needs to solve similar issues. Thanks to the DEB API interface, the applications are able to easily share also the data, thus extending the research possibilities. And finally, since the publisher for both dictionaries is OUP, the format and tools implemented for the publication of results may be reused.

7.2 TeCU – terminological thesaurus methodology

The ongoing project (2014–2015) for the Czech Office for Surveying, Mapping and Cadastre aims to create an application and methodology for semiautomated building of terminological
Specialists in every field of work use their own domain-specific vocabulary and it is desirable to share the same terminology amongst the professionals. Detailed domain terminology is not usually included in general language dictionaries, thus specialized terminological dictionaries are needed. With the need to share information unambiguously in different languages, terminological dictionaries link original terms to their translations. The taxonomical ordering of the terminology is described by term relations such as synonymy, or hyperonymy and hyponymy. The information is presented and visualized in a way that helps the readers (both specialists and general public) understand the term meaning and usage in contexts. If the data are encoded properly, the system enables automatic processing and integration of the data in third-party applications.

Natural language is ever evolving and new words keep appearing or the usage and meaning of words change. This evolution is even more noticeable in specialized vocabularies. The thesaurus system thus may employ sophisticated methods of detecting emerging words and distinct new terms in the given domain by processing synchronous domain-oriented corpora. The Natural Language Processing Centre in cooperation with the Czech Office for Surveying, Mapping and Cadastre is developing a system for building and extensions of specialized terminological thesaurus for the domain of land surveying and land cadastre. The project consists of two interconnected parts – an application to create, edit, browse and visualize the terminological thesaurus, and the tools to build large corpus of domain oriented documents with the possibility to detect newly emerging terms, or terms missing from the thesaurus. Already available tools for corpus building and term extraction and the DEB platform are utilized. During this project, we are enhancing the corpus tools (mainly to support parallel multilingual corpora), building the thesaurus web application (not limited to single domain), and developing methods to inter-connect the domain corpus with the terminological thesaurus.

Currently, the first phase of the project is finished. We have built the multilingual corpora of land surveying oriented documents and we are able to detect domain specific terms. A multi-platform web-based editor and browser application has been built, based on the DEB platform. Although this project aims to build and manage the terminological thesaurus of land surveying domain, the tools may be reused for any other domain dictionary, thus stimulating the sharing of information and general awareness of the selected domain.

**Specialized Corpus and Term Extraction**

To build the specialized corpus for land surveying and geoinformation domain, we have followed the principles designed for creation of large corpora extracted and processed from web
Table 7.3: Thesaurus size statistics.

<table>
<thead>
<tr>
<th></th>
<th>Total number of terms</th>
<th>English translations</th>
<th>German translations</th>
<th>Slovak translations</th>
<th>Russian translations</th>
<th>French translations</th>
</tr>
</thead>
<tbody>
<tr>
<td>total number of terms</td>
<td>8,783</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hyponymic relations</td>
<td>10,020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>meaning explanations</td>
<td>4,124</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The data for the corpus were gathered from publicly available online resources utilizing two different methods developed by the NLP Centre. We have built specialized corpora of land surveying domain for Czech, English, German, French, Russian, and Slovak language.

Firstly, a set of main websites related to the land surveying, the cadastre of real estates, and related topics was enlisted. Secondly, based on the content of these root websites a broader set of documents from 1,063 websites utilizing the WebBootCat tool [BKPR06] was obtained. This method needs a set of “seed words” to search the web for relevant documents. We used the main domain terms obtained from existing publicly available terminological dictionary [Há12] as seed words. The resulting corpus is used for extraction of new suggested terms for inclusion in the thesaurus.

Non-textual and low quality content was removed from the downloaded documents, utilizing the Justext tool [Pom11]. Subsequently, duplicate documents or parts (e.g. paragraphs) of the documents were purged with the Onion tool [Pom11].

Following the corpus creation, a list of “candidate terms” (proposals to include into the thesaurus) was prepared. The candidate terms were extracted from the specialized land surveying and geoinformation corpus by employing the process of corpora comparing and keywords extraction (the process is described in detail in the following papers [Kilo1, Kilo9]). Frequencies of words and named-entities in the specialized corpora are compared to the frequencies of the same phrases in a general language corpus. The best candidate terms have the highest frequency quotient [KJK+14].

Thesaurus Building

Although the main aim of the thesaurus development is to publish the authorized specialized terminology and its updates both to the experts, and general public, the thesaurus will contain broad vocabulary of related terms. Users may search even for unofficial terms and thanks to the relations between the terms and the detailed information on the source of given term, the user will find the related terms and links to the recommended official variant.

To build the thesaurus covering broad domain vocabulary, several resources are combined. In the first stage, the current authorized terminological dictionary [Há12] (containing almost 4,000 terms’ definitions and translations, but not offering the taxonomy network) was com-
Figure 7.2: Browsing the thesaurus, with detailed information for one term.

Combined with the hypero/hyponymic tree of over 6,800 entries (containing hyponymic relations, but no detailed information about terms) and 450 candidate terms extracted from the domain corpus.

The first two resources were available in HTML form, tagging parts of entry structure, but still leaving a lot of text in an unstructured format. It was necessary to tidy up the data and convert resources to the unified XML format for the database storage. In the case when the terms were shared by both dictionaries, combined term entries were created, containing both detailed information on terms, and the term relations. See Table 7.3 for more details regarding the current size of the thesaurus. Resulting thesaurus was extended by including the suggestions of the terms automatically extracted from the corpus. These suggestions are inserted in a separate category, together with the information regarding the source and reliability of the term.

In the next stage, the thesaurus will be expanded even more by including several resources:

- appropriate parts of the GEMET4 (General Multilingual Environmental Thesaurus),
- regularly updated Registry of Territorial Identification (RUIAN)5,
- automatically extracted multi-lingual terms,
- suggestions from the public users.

**Editing Tool**

The thesaurus editing tool is implemented as a client-server application, with the DEB server providing the database and management back-end. The client-side application is a multiplatform web application accessible in any modern browser, based on open-source technologies –

4http://www.eionet.europa.eu/gemet
5http://www.cuzk.cz/ruian/
Figure 7.3: Entry relations visualized.

Figure 7.4: Editing the term entry.
JQuery\(^6\) and SAPUI5\(^7\) libraries for graphical interface. The client and the server communicate using standardized interface over HTTP with the data encoded in JSON format.

The standardized application interface also allows an integration of third-party applications that reuse the thesaurus data. The interface provides all the functions needed to work with the data (e.g. search queries, browsing the thesaurus structure and detailed entry information, entry creation and updates...). Two methods to utilize the interface are available supporting modern web-service standards – REST/JSON [FT00] and WSDL [CCMW01]. One of the intended use-cases is the integration into the Geoportal\(^8\), where the terms are to be used for the document metadata and categorization.

The thesaurus web application itself provides a graphical interface for browsing the hyponymic tree (see Figure 7.2). Out of the several possible visualizations of the tree, the expanding multi-level tree was selected. Although it may not display all the relations in a proper graph form, it is much more intuitive for the users (based on the experience with the DEBVisDic system). Terms with several hyponyms are displayed in multiple trees in the tree structure. To further visualize the relations of a single term, a graph of hypernyms, synonyms, and other related terms is displayed (see Figure 7.3).

A detailed description is given for each term, including meaning explanation, translations, or accepted variants (see Figure 7.4 for a sample of editing form). When more sources are incorporated in the thesaurus, the reliability of each source and revision history will be presented to the users. Source reliability follows the rating scale of the Office for Surveying – the most reliable are terms authorized by the terminological committee, followed by terms used in scien-

\(^6\)http://jquery.com/
\(^7\)https://sapui5.netweaver.ondemand.com/
\(^8\)http://geoportal.cuzk.cz/
entific journals, and the terms made up by general public at the bottom of the scale. Users or third-party applications may decide which sources or terms they prefer to work with.

To get a better picture of a term and its usage, extended information from the corpus are presented. Users may consult full examples (see Figure 7.5) or related words from the corpus (see Figure 7.6).

Second phase of the project

In the next phase of the project, multi-lingual and multi-source aspects of the thesaurus will be extended. We have already built corpora in several languages and we will provide automatically extracted terminology from these corpora as the suggestions for terminology translation.

Hand in hand with adding more sources for the thesaurus terms, the editing and browsing application will offer options for filtering the terms based on the source reliability and authorization status and periodic semi-automatic imports from authorized sources.

7.3 Dictionary of the Czech Sign Language

Preparation of the new dictionary of the Czech Sign Language is one of the key activities of the Network of Expert Centres Providing Inclusion in Tertiary Education project. The dictionary is created in co-operation with several Czech universities and organizations dedicated to the sign language issues, and co-ordinated by the Teiresiás Centre, Masaryk University. At the current

---

*Officially named Support Centre for Students with Special Needs.*
stage of the process, the aim is to create an extensive dictionary of the Czech sign language and the explanatory dictionary of the Czech language, both inter-connected to serve also as a bilingual dictionary. More languages (both sign and spoken) may be added later to form a multilingual dictionary. The paper regarding the software system for the dictionary editing is to be published in the proceedings of the NLDB 2015 conference [RH15].

Initially, the dictionary data were edited in a custom web application, which was aimed more at the website content management than dictionary writing. However, the application did not support all the requirements of the project. After reviewing several options, the DEB platform was selected to develop the dictionary writing system. An editing application is implemented using the ExtJS library [Sen15]. Public dictionary browser is a standard web application, enhanced with the jQuery library [jQu15a], mainly to enable the user personalization, and the Flowplayer [Flo15] library for the video playback.

**Entry representation**

Creating dictionaries of sign languages has always been challenging in printed form, this may be changed with the electronic dictionaries. Kristoffersen and Troelsgard have described their experience from building the Danish Sign Language Dictionary in [KT12]:

> The overriding challenge in sign language lexicography is how to render signs in the absence of a written language. ... the use of video recordings has predominated. This approach is obviously limited to electronic dictionaries.

The Czech Sign Language dictionary relies heavily on video recordings, which are used not only for the sign representation (front view and side view), but also for meaning explanations or usage examples. See Figure 7.8 for the example of an entry in the Czech Sign Language dictionary. However, the video is not the only way to present a sign to the dictionary users. The dictionary also provides tools to enter and display two methods of formal sign transcription (see Figure 7.7 for comparison).

- **SignWriting** [Suto09, Kat08] is an iconic script system in which the hand shapes, movements and other sign language features are encoded with a set of image symbols. To edit

---

10 Dictionary is already publicly available at the [http://www.dictio.info](http://www.dictio.info).
Figure 7.8: Entry in the Czech Sign Language dictionary.
the SignWriting for a dictionary entry, authors may either use the specialized tool to work with the icons directly, or enter a standardized string representation. The dictionary system also handles correct SignWriting composition, e.g. for collocations. However, the composition of signs is not straightforward. Collocations are usually shown as a series of single signs, but the movements and transitions between signs are smooth. This may change the presented sign a little and it is represented by several SignWriting symbols. The dictionary system has to select correct symbols when combining the signs.

- Hamburg Notation System (HamNoSys) [Han04] is a transcription system, similar to IPA [Int99] for spoken languages. The main advantage of HamNoSys is the formal syntax of notation language, making the notation more useful for computer processing. For example, HamNoSys transcription is used in the avatar visualization tools. In the dictionary, authors may use a specialized tool to enter the notation symbols.

Access control and validations

The dictionary is produced by several teams at different institutions, each covering different parts of a dictionary entry or different domain. Users may get various roles (e.g. editor, reviewer, supervisor...) and the DEB platform access management module combines user roles with groups, resulting in complex access control system. Furthermore, each part of a dictionary entry (e.g. grammatical information, definitions, translations...) has to be approved by the supervisor before it may be published for public presentation.

Validations are also needed to check the completeness and correctness of input information for each dictionary entry. The first stage of validations is provided during the editing, for example correct references between the sign language and the Czech language. More validations are run after a batch of entries is handed over by a group of authors to the reviewers. During the review phase, reviewers may combine several conditions to form complex validation queries. For example, the reviewer may check for entries that contain video recording of the lemma and the definition, but do not contain the usage example or translations. The possibility to run complex validations after each phase is essential for the successful completion of the dictionary.

Data import

Each institution or department involved in the project specializes on one task. For most of the entries in the initial phase, the following process was employed. One department recorded the videos for a batch of signs. The batch was then handed over to the lexicographic team that added grammatical information, translations, etc. Meanwhile, another team provided SignWriting transcriptions. To speed up the creation of new entries and updates of existing entries, the dictionary system supports complex tools for batch import of video recordings.
In the post-processing, all video files are named according to a standardized methodology. The import system scans the batch of video recordings and detects the type of a video (sign, definition, usage example) and checks whether the file is an update for existing video, a completely new entry, or an additional video for an existing entry. The system is able to detect whether to add a definition video to a certain entry meaning. Translations to the Czech language are proposed, linking with existing Czech entries or creating new Czech equivalents. Before updating the database, users may review and fix the inconsistencies if needed.

As for the Czech language dictionary part of the project, the goal was not to create a completely new dictionary, but to reuse and update existing resources. For that reason, three large monolingual dictionaries were included into the system: The Academic Dictionary of New Words [KP+99], The Dictionary of Written Czech [F+95], and The Dictionary of Czech Neologisms [M+04]. Each of the dictionaries follows different entry structure. During the import, all the information had to be normalized to a common entry structure while not losing the information.

To enrich the dictionary entries for the Czech language with "real world" usage examples, the dictionary writing system provides examples extracted from the corpus (in this case, CzTenTen12 corpus [Suc12]). Using the SketchEngine API, the DEB editing application displays a set of the most suitable examples (as rated by the GDEX tool [RHK+08]). The authors may select the best example directly in the entry editor without the need to browse the corpus.

**Searching**

Searching in a sign language dictionary is a challenging issue. For the Czech Sign Language dictionary, several methods are provided and users may choose their preferred method. Thanks to the cross-references between sign language and spoken language dictionaries, users may search for a word in a written form and the sign language translations are displayed. Experienced users...
may enter the SignWriting or HamNoSys transcription (with the help of the same tool used to enter transcription by the authors).

The most convenient search method is the "iconic search". Users may select hand shapes or positions from the set of images. The selection is internally converted to SignWriting representation. Users are able to enter even very complex queries just by clicking on a few icons. The graphical interface for this search method is inspired by the Dutch Sign Language Dictionary [KT12], but provides more options to exactly specify the signs (e.g. sign symmetricity, or hand positions). See the shape selection in Figure 7.9.

**Personalization**

The aim of the dictionary is to include as much information as possible, however, that may be overwhelming for some users. For example, HamNoSys notation usually takes quite a lot of space, but most users are not able to read it. Users of the published dictionary are able to set preferences and decide which information they want to see in the dictionary. They may also choose a preferred main entry representation, ie. what is the main way to present signs in lists or cross-references (video preview, SignWriting, or HamNoSys).

**Project results**

The dictionary writing system was successfully implemented and tested, the dictionary editing is currently ongoing. The dictionary statistics are provided in Table 7.4

<table>
<thead>
<tr>
<th>Table 7.4: Czech Sign Language Dictionary statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>videos, front view</td>
</tr>
<tr>
<td>videos, side view</td>
</tr>
<tr>
<td>videos, explanations</td>
</tr>
<tr>
<td>videos, usage examples</td>
</tr>
<tr>
<td>videos, total</td>
</tr>
<tr>
<td>entries, Czech Sign Language</td>
</tr>
<tr>
<td>entries, Czech</td>
</tr>
</tbody>
</table>
It can work with links only between synsets which is a limiting feature for enriching wordnets with various sorts of information, e.g. in Czech with word derivation relations existing within one part of speech as well as across them.

The first step of the DEBVisDic development was the reimplemention of the VisDic semantic network editor within the DEB platform, which was subsequently extended with new features supporting the linguistic work on wordnets.

DEBVisDic architecture

DEBVisDic client software was initially developed using the Mozilla Cross Platform Engine (see chapter 6.2.2 for more information), and is co-authored by Martin Kudlej, see for example [RK06].

Although DEBVisDic is used mostly for wordnet editing, it is designed according to principles that allows easy interconnection of various lexical resources (e.g. semantic networks, or ontologies). To follow this design, DEBVisDic is split into several modules, instead of one complete software package, thus following the modular structure of the DEB platform server part. DEBVisDic core with shared functionality is the main module, and each wordnet (or other lexical resource) is presented as a separate module. Because the DEBVisDic was implemented using the Mozilla platform technology, each module is represented as a separate extension for Firefox (or other Mozilla-based) browser. While it may seem unnecessary to have separate modules for each resource, when most of them are wordnets with similar structure, this approach has two main advantages. Firstly, such application design greatly supports extensibility and document structure changes for various resources – both small changes for various national wordnets, and...
DEBVisDic – synset preview.

Secondly, users do not need to install all the modules, they only download modules for the resources they work with.

**DEBVIsDic features**

DEBVIsDic introduced a new versatile interface (see the Figure 7.10) that allows the user to arrange the work without any limitations.

Users are able to open multiple views of multiple resources to compare them or to help with creating new data. Standard wordnet module supports several types of data views:

- textual preview of the data (see Figure 7.11), where it is also possible to have several different preview layouts (e.g. basic and detailed information), and users may customize the view (e.g. colours);
- plain XML view, showing the data as stored in the database (see Figure 7.12);
- hypero-hyponymic tree, representing the structure of semantic network.

Since the semantic network is not just a simple tree structure, but a directed graph structure, the DEBVIsDic displays two types of hypero-hyponymic trees to represent the data better for users. The default tree view uses the top synset of the semantic network as a tree root and shows the path to the selected synset (and its hyponyms). The other view is called "reverse tree" and uses the selected synset as a tree root and displays the path to the top synset. See Figure 7.13 for comparison.
Users may use the hypero-hyponymic tree to navigate the wordnet and to find synsets. Another option to find synsets is the search query. DEBVisDic client supports simple queries (e.g. searching for a word in synonyms), and also complex queries for advanced users. It is possible to enter specific elements or attributes of the data XML structure with the complex queries, and to compare values from different wordnets. A complex query language is used by the DEBVisDic to provide inter-dictionary links, for example selecting the synsets that represents specific concept from the ontology. When users click on the link in synset preview, it is translated to the complex query to filter the results.

It is possible to edit the synsets (if the user has access rights for editing, of course). When the newly created wordnet is based on an existing one, users may copy the information from an existing wordnet. Users may either copy full synset information to the new wordnet and edit them, or just create link between the wordnets.

During the work on adding new items to the wordnet ontology, users often consult different
ontological resources. Including specific parts of these ontologies into wordnet often works with the same hierarchy as it is defined in the source ontology.

For such cases, DEBVisDic provides a technique for efficient saving of several synsets and their hierarchical structure in one step, using the defined API interface. With this API call, DEBVisDic can store several synsets at once, while keeping their defined structure. Of course, before saving the synsets, the user does not know the unique synset database identifiers for new synsets. To be able to define the synset hierarchy, the user uses temporary identifiers in the request and the DEBVisDic server replaces them with real identifiers.

When the users work with several wordnets that are connected together (usually by linking to one "pivot" wordnet), it is possible to synchronize several views. The synchronization is either automatical, or on user request, but the technique is the same. When users select the specific synset (e.g. in the hypero-hyponymic tree) in one of the synchronized wordnets, corresponding synsets are displayed in other wordnets. The synchronization is defined by complex search queries, so it is possible to link various resources, for example, synchronize the ontology concepts with wordnet synsets.

For complex multi-language projects like Global WordNet Grid (described below) or KYOTO (described in chapter 7.6), different wordnets may be connected not just through one pivot wordnet, but through several center wordnets.

There are several types of external relations used in inter-wordnets links. The most common relations are: synonym, near synonym, has hyperonym, and has hyponym. DEBVisDic API functions allow to quickly find all the synsets from several wordnets that are related with the pivot synset.

Utilizing the DEB platform features, DEBVisDic keeps track of the editing process, making sure that authors do not overwrite each others work, and journaling the data updates. It is also possible to reuse the DEB platform modules and extend DEBVisDic with more resources. For example, the Czech wordnet is interconnected with the data from the VerbaLex valency lexicon [HH05].

**DEBVisDic for the Global WordNet Grid**

It was proposed [HPR08] at the Global Wordnet Conference to use DEBVisDic as a tool to support the Global WordNet Grid. The Grid is a network of freely available wordnets in many languages built around a shared set of concepts [PFV08].

In the DEB platform environment, all the wordnets are usually stored on single DEBVisDic server. In the Grid, most of the wordnets will be also stored in this way, however, since the Grid will be finally composed of a large number of wordnet dictionaries developed by different organizations, this solution may not always be the best option (for example because of licensing
issues). Thanks to the client-server nature of the DEB platform, DEBVisDic can offer two possible types of encapsulating wordnets in the server:

- A wordnet can be physically stored on the central server. This is the traditional DEBVisDic setup and offers the best performance.

- A wordnet can be stored on a DEBVisDic server located at the wordnet owner’s institution. All servers in the Grid can then communicate with each other (depending on the server setup). The Central Grid server for this wordnet has only the knowledge of which server to contact, instead of having the full wordnet database stored locally, and all queries are dynamically resolved over the Internet. This option may be slower as it depends on the quality of connection to different servers and their performance. On the other hand, the wordnet owner has full control over the displayed data and access permissions.

- A mixed solution – selected wordnets are stored on central server, while others are stored on their respective owners’ servers. This is just an extension of the previous option. Again, the performance of the whole Grid depends on the performance of single servers, but the speed can be improved if the most used wordnets are stored on the central server.

The DEB platform provides several possibilities of working with the wordnet data. All types of the Grid access undergo the same control of service and user management with the option to provide information for public (anonymous) access as well as authenticated access for registered users. Basically, each wordnet in the Grid can be presented to the Grid users in one of the following forms:

1. by means of a simple purely HTML interface working in any web browser. This interface is able to display one wordnet dictionary or the same synset in several wordnets.
Synsets are displayed using XSLT templates – the server can provide several views of the synset data ranging from a terse view up to a detailed view. The view can be even different for each dictionary. An example of such presentation of one synset in three wordnets is displayed the Figure 7.14. This type of wordnet view is probably the best for public anonymous access to the Grid.

2. using the full DEBVisDic application with the features described earlier. With this type of the Grid access, the user has the most advanced environment for working with the Grid wordnets.

3. by means of a defined interface of the DEBVisDic server. This way any external application may query the server and receive wordnet entries (in XML or other form) for subsequent processing. In this way, local external applications can easily process the Grid data in standard formats.

DEBVisDic 2 – new web application

As discussed in chapter 6.2.2, client applications initially developed as Firefox extensions have been recently reimplemented as web applications, including DEBVisDic. New client software was co-authored by Tomáš Hrušo and was presented in the proceeding of the RASLAN conference in paper [RH13].

Thanks to the client-server architecture of the DEB platform, no changes were needed on the server side. Only the client side application needed to be reimplemented, reusing the existing DEB interface. Main feature requests when designing the new version were to keep all DEBVisDic features, and to provide application working in all major web browsers.

Similar to the previous version, DEBVisDic 2 aims primarily on wordnet-type semantic network browsing and editing, but supports various types of dictionaries. The application consists of the main window with settings and separate windows for each dictionary that a user wants to edit, following the design of separate Firefox extensions for each dictionary from DEBVisDic. A single dictionary window includes a list of entries (synsets) and a set of tabs with several views on selected entry: basic preview, XML representation, hypero-hyponymic tree, and editing form. Context (right-click) menu provides functions for displaying and creating inter-dictionary links (e.g. display all synsets using selected ontology term).

DEBVisDic 2 utilizes Model-View-Controller architecture, and the design follows this principle. Current open standards are used in the application: HTML and CSS for data presentation (view), and JavaScript for application logic scripting (model, controller). The application is modular, with separate core shared by all the dictionaries, and a plugin with specific functionality for each type of dictionary. In most cases, DEBVisDic is used for wordnet editing, thus the default wordnet module is provided unless the application specifies otherwise.

After reviewing several JavaScript frameworks, it was decided to use jQuery library [jQuerya],

80
for the DEBVisDic 2 development. One of the most challenging features was the implementation of the context menu functions, because of the huge differences in various browsers. In the end, the context menu behaves the same as in the original DEBVisDic application, with the help of jQuery contextMenu plugin\(^1\). Pretty printing of entry in XML format is provided by the Prettify plugin\(^2\).

Apart from complete reimplementation of the DEBVisDic application, the new version introduced several new features, for example, saving user settings (e.g. opened dictionaries and window positions) on the server, thus allowing user to switch browsers and computers, and continue in work.

Another major new feature are more generalized links and relations between dictionaries. It is possible to use any part of an XML entry to build inter-dictionary search queries, for example, selecting all synonyms in a synset, automatically showing details of ontology term for selected synset, or all equal synsets between two wordnet languages.

At the time of the DEBVisDic 2 development, the project started for the creation of Open Dutch WordNet \([PV14]\), based on the data from the Cornetto project (see chapter 7.5). Since the Cornetto software is developed as a set of modules for the DEBVisDic, *Cornetto 2* modules were also reimplemented together with the DEBVisDic 2. Because of the database design, specific dictionary modules were needed for Cornetto Synsets, Cornetto Lexical Units, Open Dutch WordNet, and English Wordnet, all inter-connected together.

Lexicographers of the Open Dutch WordNet team were heavily testing the DEBVisDic 2 application during the development and provided valuable feedback. Since the autumn of 2013, DEBVisDic 2 is used intensively for the Open Dutch WordNet editing. The first version of the Open Dutch WordNet was released in the December 2014, containing 116,992 synsets. DEBVisDic 2 browser is provided as a public interface to browse the wordnet\(^3\).

At the beginning of 2014, the editing and extension of the South African wordnets also switched from the original DEBVisDic application to the DEBVisDic 2. Subsequently, the DEBVisDic 2 was released for public use.

DEBVisDic 2 is not only more user-friendly, especially because of the accessibility in all major web browsers and no need to install the software, but also easier to manage on the server side. Utilizing the new features and application design, a new service was introduced as a part of the DEB platform. Users are now able to automatically create new wordnet and edit it in DEBVisDic 2. Wordnet authors may decide to share the wordnet with other users, selecting their access rights (read-only, or editing), or even share the wordnet with public. This service is currently in the public testing and we hope that it will encourage the creation of more national wordnets and other lexical resources.

---

\(^1\)http://medialize.github.io/jQuery-contextMenu/
\(^2\)http://google-code-prettify.googlecode.com/svn/trunk/README.html
\(^3\)http://wordpress.let.vupr.nl/odwn/demo/
Cornetto\textsuperscript{14} was a project by the Vrije Universiteit Amsterdam and the University of Amsterdam to build a lexical semantic database that combines wordnet with FrameNet-like information \cite{FBS04} for Dutch. The combination of the two lexical resources results in a much richer lexical database that may improve natural language processing (NLP) technologies, such as word sense-disambiguation, and language-generation systems. In addition to merging the wordnet and FrameNet-like information, the database is also mapped to a formal ontology to provide a more solid semantic backbone. Cornetto development was presented in several conference papers, e.g. \cite{HVR08a, HVR08b}.

The database is filled with data from the Dutch Wordnet \cite{Vos98} and the Referentie Bestand Nederlands \cite{MMdM99}. The Dutch Wordnet (DWN) is similar to the Princeton Wordnet for English, and the Referentie Bestand (RBN) includes frame-like information as in FrameNet plus additional information on the combinatoric behaviour of words in a particular meaning.

Both DWN and RBN are semantically based lexical resources. RBN uses a traditional structure of form-meaning pairs, so-called Lexical Units \cite{Cru86}. Lexical Units contain all the necessary linguistic knowledge that is needed to properly use the word in a language. The Synsets are concepts as defined by \cite{MF91} in a relational model of meaning. Synsets are mainly conceptual units strictly related to the lexicalization pattern of a language. Concepts are defined by lexical semantic relations. For Cornetto, the semantic relations from EuroWordNet were taken as a starting point \cite{Vos98}. The aim of the project was to clarify the relations between Lexical Units and Synsets, and between Synsets and an ontology.

The Cornetto database (CDB) consists of 3 main data collections:

- collection of Lexical Units, mainly derived from the RBN;
- collection of Synsets, mainly derived from DWN;
- collection of Terms and axioms, mainly derived from SUMO and MILO.

In addition to the 3 data collections, a separate collection of so-called Cornetto Identifiers (CIDs) is provided. These identifiers contain the relations between the lexical units and the synsets in the CDB, and also to the original word senses and synsets in the RBN and DWN. See Figure 7.15 for a schema of Cornetto data collections.

DWN was linked to WordNet 1.5. WordNet domains are mapped to WordNet 1.6 and SUMO is mapped to WordNet 2.0 (and most recently to WordNet 2.1). In order to apply the information from SUMO and WordNet domains to the synsets, mapping tables between the different versions of Wordnet were used. We used the tables that have been developed for the MEANING project \cite{WNM07, DJG03}. For each equivalence relation to WordNet 1.5, we consulted a

\textsuperscript{14}Combinatorial and Relational Network as Toolkit for Dutch Language Technology
table to find the corresponding WordNet 1.6 and WordNet 2.0 synsets, and via these we copied the mapped domains and SUMO terms to the Dutch synsets.

The structure for the Dutch synsets thus consists of:

- a list of synonyms,
- a list of language internal relations,
- a list of equivalence relations to WordNet 1.5 and WordNet 2.0,
- a list of domains, taken from WordNet domains,
- a list of SUMO mappings, taken from the WordNet 2.0 SUMO mapping.

The structure of the lexical units is fully based on the information in the RBN. The specific structure differs for each part of speech. At the highest level it contains:

- an orthographic form,
- morphology,
- syntax,
- semantics,
- pragmatics,
- examples.

DEBVisDic was specifically adapted for the purpose of the project. A separate editing module was created for each of the data collections described above. Cornetto client software was co-authored by Martin Kudlej in the original version, and later reimplemented as a part of the DEBVisDic 2 development (see section 7.4), co-authored by Tomáš Hrušo.
The user interface is the same as for all the DEBVisDic modules: the upper part of the window is occupied by the query input line and the query result list and the lower part contains several tabs with various views of the selected entry. Searching for entries supports several query types – basic query is to search for a word or its part, the result list may be limited by adding an exact sense number. For more complex queries, users may search for any value of any XML element or attribute, even with a value taken from other dictionaries (the latter is used mainly by the software itself for automatic lookup queries).

The tabs in the lower part of the window are defined per dictionary type, but each dictionary contains at least a preview of an entry and a display of the entry XML structure. The entry preview is generated using XSLT templates, so it is very flexible and offers plenty of possibilities for entry representation.

**Cornetto Lexical Units**

The Cornetto foundation is formed by Lexical Units. Each entry contains complex information about morphology, syntax, semantics and pragmatics, and also lots of examples with complex substructure. Thus one of the important tasks was to design a preview to display everything needed by the lexicographers without the necessity to scroll a lot (see Figure 7.16). The examples were moved to separate tabs and only their short resumé stayed on the main preview tab.

Lexical units also contain semantic information from RBN that cannot be published freely because of licensing issues. DEBVisDic needs to differentiate the preview content based on the access rights of each user.

The same ergonomic problem had to be resolved in the edit form. The whole form is divided into smaller groups of related fields (e.g. morphology) and it is possible to hide or display each group separately. By default, only the most important parts are displayed and the rest is hidden.
Another new feature introduced for the Cornetto system is the option to split the edited entry. This function duplicates the content of an edited entry to create a new entry. This way, users may easily create two lexical units that differ only in selected details.

Because of the links between all the data collections, every change in lexical units has to be propagated to Cornetto Synsets and Identifiers. For example, when deleting a lexical unit, the corresponding synonym has to be deleted from the synset dictionary.

Cornetto Synsets

Synsets are even more complex than lexical units, because they contain lots of links to several resources – links to lexical units, relations to other synsets, equivalence links to Princeton English WordNet, and links to the ontology.

Again, designing a user-friendly preview containing all the information was very important. Even here, the preview needed to be split into two tabs – the first with the synonyms, domains, ontology, definition, and short representation of internal relations, and the second with full information on each relation (both internal, and external to English Wordnet), see Figure 7.17. Each link in the preview is active and displays the selected entry in the corresponding dictionary window. For example, clicking on a synonym opens a lexical unit preview in the lexical unit window. Users may also display detailed information for all lexical units that form a synset, see Figure 7.18.

The synset window also offers a tree view representing a hypernym/hyponym tree, together with the reversed tree displaying relations in the opposite direction (as used in DEBVisDic, see section 7.4). The tree view also contains information about the significance of each subtree – e.g. the number of direct hyponyms or the number of all descendant synsets.

The synset edit form is similar to the form in the lexical units window, with less important parts hidden by default. When adding or editing relation links, users may use the same queries.
as in dictionaries to find the right entry.

**Cornetto Identifiers**

The lexical units and synsets are linked together using the Cornetto Identifiers (CID). For each lexical unit, the automatic aligning software produced several mappings to different synsets (with different score values). At the very beginning, the most probable one was marked as the “selected” mapping.

In the course of work, users have several ways of confirming the automatic choice, choosing from other offered mapping, or creating an entirely new link. For example, a user may remove an incorrect synonym from a synset and the corresponding mapping will be marked as unselected in CID. Another option is to select one of the alternate mappings in the Cornetto Identifiers edit form. Of course, this action leads to an automatic update of synonyms.

The most convenient way to confirm or create links is to use *Map current LU to current Synset* function. This action can be run from any Cornetto client package, either by a keyboard shortcut or by clicking on the button. All the required changes are checked and carried out on the server, so the client software does not need to worry about the actual actions necessary to link the lexical unit and the synset.

**Cornetto Ontology**

The Cornetto Ontology is based on SUMO and the editing module was developed by enhancing the SUMO editing module. The ontology is used in synsets, as can be seen in the Figure 7.17. The synset preview shows a list of ontology relations triplets – relation type, variable, and variable or ontology term. Clicking on the ontology term opens the term preview. A user can also browse the tree representing the ontology structure.
Project results

The Cornetto lexical database was built during the STEVIN project STE05039 in 2006–2008 and resulted in a database containing 70,371 synsets and 119,108 lexical units. The lexical database was later used and extended as the Cornetto 2.0 during the DutchSemCor project in 2009.

Because the work with the Cornetto data still continues, now as part of the creation of Open Dutch WordNet, the editing modules were reimplemented together with the development of the DEBVisDic 2 application (see section 7.4).

7.6 KYOTO – ontologies for knowledge detection

KYOTO\textsuperscript{15} was a project co-funded by the EU FP7 programme, carried out by several research organizations from Europe and Asia, and lead by the Consiglio Nazionale delle Ricerche. The goal of the project was a system that allows people in communities to define the meaning of their words and terms in a shared platform so that it becomes anchored across languages and cultures but also so that a computer can use this knowledge to detect knowledge and facts in text. The description of the DEB platform integration to the KYOTO system was published e.g. in the [HR10, HR09].

WordNet semantic networks allow to express basic language relations in a multigraph structure directly processable by computer systems. However, a description of more complicated structured knowledge, e.g. relations with more than one participants, cannot be encoded in a WordNet-standard way for further computer analysis.

In the KYOTO system, this (potential) drawback of WordNet is solved by the idea of extending the WordNet into a \textit{WordNet Grid} of multiple languages with a shared ontology in the center. Interlinking national wordnets is not a new idea, it was introduced e.g. in the EuroWordNet [Vos98] and Balkanet [Chr04] projects. In these projects the “pivot,” i.e. the \textit{interlingual index}, was represented directly by the English WordNet. This solution had several advantages and several disadvantages. From the point of view of the knowledge analysis, the biggest disadvantage was that the lexical knowledge structure was “hidden” in the English lexicon without the possibility to really extract it for the purpose of further computer processing. The shared ontology provides a way of adding structural semantics to the interlingual links.

It is a known fact that, for instance, the results of EuroWordNet are not freely accessible though the participants of the project have developed (and are developing) more complete and larger wordnets for the individual languages. Practically the same can be said also about the results of the Balkanet project. If one wants to exploit wordnets for different languages it is almost always necessary to get in touch with the developers and ask them for the permission to use the wordnet data.

\textsuperscript{15}Knowledge Yielding Ontologies for Transition-based Organization
Another reason for building and having the completely free wordnet Grid is the fact that the particular wordnets can be linked to the selected ontologies (e.g. SUMO/MILO) and domains. This has already took place with the wordnets developed in the Balkanet project. The KYOTO project incorporated and expanded the Global Wordnet Grid and is the first system that exploits the benefits of storing the definitions of terms and facts in a computer processable logical system using the Grid’s shared ontology.

Because DEBVisDic was successfully used to build several wordnets and other semantic networks (e.g. in the Cornetto project, described above), the DEB platform was selected to be one of the central parts of the KYOTO system. The DEBVisDic server modules are serving as a storage database and backend for the ontologies, wordnets and other knowledge resources used in the KYOTO system (see Figure 7.19).

For the purpose of the KYOTO system, a specific user interface was designed, so the DEBVisDic client application was not used for the semantic network editing. Instead, a custom interface was built based on the wiki platform that utilizes the DEBVisDic server application interface.

During the development of the KYOTO system, several new features were introduced to the DEBVisDic and the DEB platform in general that are described in the following text.
Table 7.5: Translating API call – translate bosque (forest) from Spanish to Japanese

Query: https://server_name/wnspa?action=translate&query=bosque &target=wnjpn

Result:

```
{
  "translated": [
    {
      "elr": ["eng-30-09284015-n"],
      "value": "jpn-09-09284015-n",
      "label": "[n]森林地;森林;森林地;森林;森林;林地:"
    },
    {
      "elr": ["eng-30-08438533-n"],
      "value": "jpn-09-08438533-n",
      "label": "[n]森林;森林;森林;森林;林地:"
    }
  ]
}
```

Translate Synsets

One of the main advantages of wordnets is their multilinguality, i.e. the design of interlingual index used as a pivot between several national wordnets. Currently, the most commonly used pivot is the English wordnet due to its size, completeness and good maintenance. The link to the pivot is encoded either by assigning the English wordnet synset ID directly to the national synset, or by means of external relations (ELR) as specific “pointers” to other synsets outside the actual dictionary.

A common operation in multilingual projects is thus “translating” a word (in all synsets) to another language by means of the selected pivot. Since the mapping from national wordnets to English wordnet is not always unambiguous, each synset can point to one or more synsets in common wordnet. The DEBVisDic API method thus provides all possible synsets in the target language.

For instance, when we take the word bosque (forest) in Spanish, we can find this word in two synsets bosque:1 and bosque:2 that are linked to forest:1, wood:2, woods:1 and forest:2, woodland:1, timberland:1, timber:4 in the English wordnet. Through the obtained English IDs, direct equivalents are enlisted e.g. in the Japanese wordnet – see the example in Table 7.5.

Links between Wordnets and Ontologies

All wordnets in the KYOTO database are interlinked using the common central ontology. The solution is not limited to one ontology only, and different domain ontologies can extend the information for some synsets. Apart from the KYOTO Central Ontology, four different thesauri are used:
In the KYOTO system, all the ontologies are converted to the standard RDF/SKOS [MB09] format and stored in the OpenLink Virtuoso Database backend [WI09]. The Virtuoso database provides built-in support for the RDF SPARQL query language [PS09], which is designed for complex queries over ontological relations encoded in RDF triplets. Thanks to DEB platform architecture, the Virtuoso service is seamlessly integrated to all DEB interfaces, although the database itself is not part of the DEB platform. Users may enter a SPARQL query, but usually the client application prepares the query as a result from graphical user input. Subsequently, DEBVisDic server retrieves the results using the Virtuoso AJAX API and presents them to the users in the same format as other resources. See Figure 7.20 for an example of interlinking various resources together.

API Interface Enhancement

For the client application developed in the KYOTO system, several enhancements were introduced to the standard DEBVisDic API interface. These features are also available to the whole DEB platform and enable even more options to integrate the DEB platform in third-party applications and share the resources stored in the database.

References:

http://isegserv.itd.rl.ac.uk/skos/gemet/
http://www.sp2000.org
http://eunis.cea.europa.eu
Standard DEBVisDic API supported loading and saving synsets including the complete synset data. However, AJAX-based JavaScript lightweight applications frequently need to change just specific parts of the whole synset structure. Supporting this requirement, the client applications that access the wordnet data by means of the DEB application programming interface do not need to get or save all the data at once and can parse less data faster. For these reasons, the API was extended with arguments to read or write specific parts of the synset.

For security reasons, JavaScript client applications may send requests only to the server on the same domain as the application. However, when working with several services hosted on different servers (which was the case in the KYOTO system), the application needs to overcome this limitation. This kind of API requests is supported by so-called JSONP (JSON with padding) protocol. JSONP is a jQuery extension that passes the obtained server response to a specified JavaScript function. To be able to provide the results of all API calls in the form of the JSONP protocol, all the DEBVisDic API calls accept a new parameter ‘callback’. With this parameter, the response is encapsulated in the requested JavaScript function.

Project results

The KYOTO project was successfully finished in 2011. DEBVisDic database backend was used to store and access newly created wordnets for the environmental domain in seven languages (English, Dutch, Spanish, French, Italian, German, and Chinese), each containing 3,260 concepts with mappings to the ontology and extracted terminology. Domain wordnets were accompanied by already existing general wordnets for the same languages with links between general and domain wordnets. KYOTO Central Ontology was also stored in the DEBVisDic.

7.7 PRALED – Czech lexical database

PRALED application was developed for the Institute of the Czech Language (ICL) of the Academy of Sciences as a part of the research project *Creation of a Lexical Database of the Czech Language of the Beginning of the 21st Century* [RK07]. In this project, all the available resources (corpora, morphological analyzer, digitized dictionaries, ...) are used in preparing the supportive material for the future new lexicographic description of the Czech language, which will be finally published as an electronic modern monolingual dictionary of the Czech language named LEXIKON 21. Utilization of the DEB platform for the project was presented in the book [HR13].

The lexicographic work has been divided into several phases – in the years 2005–2008 all the lexicographic resources of ICL have been digitized and a new dictionary writing system (called PRALED), based on the DEB platform, has been designed and implemented. In the years
2009–2011 PRALED has been used for building a complex database of 100,000 lexical units of various types. The preparation of LEXIKON 21 will be using the results of this second phase.

The PRALED application (short for Prague Lexical Database) was developed in close cooperation with linguists and users from the Institute of the Czech language. The application was implemented using the Mozilla Cross Platform Engine.

Since the beginning of the project, the application and the user interface has been continuously updated according to the changes in the research data and the needs of the lexicographic team. Thanks to the design of the DEB platform, it was possible to prepare prototypes of new versions in short periods of time.

The design of the PRALED client and server parts is oriented to fluent editing of very complex dictionary entries. The resulting lexicographic database will contain all the morpho-syntactic information of Czech lexical entries in a machine readable form, providing an invaluable resource for both human experts as well as computer applications.

PRALED users can be divided into two groups: the ICL researchers are able to view and create entries, whereas others (usually reviewers) can only view the finished entries. A team of 25 lexicographers was using the application to create entries during the research project.

The client application consists of two parts – the entry listing, and the complex editing form. After a successful login to the application, the entry listing window is displayed. With a basic filtering, a user can search for entries by headword, or by a piece of text from the definition. In the advanced search, it is possible to freely combine any criteria from the entry data (for
example, entries edited by a selected author in January 2010). The client application translates the user selection to an XPath query that is executed on the server.

A list of selected entries is provided by the server in the RDF format. Thanks to this format, the user can view the results sorted by different fields (headword, author, last change date, etc.) and it is also possible to nest linked entries together (collocations are linked to the main headword). The resulting list can be printed in several output formats. An example of the list window is displayed in Figure 7.21.

A separate window with a preview of all information of a lexical entry is opened for each edited headword. Users are able to show or hide the data as they need for the given task. The entry window corresponds to the dictionary XML structure and is divided into general entry information and specific information for each sense. Linguists in ICL usually concentrate on one feature of a word and several people together work on each entry. To make this task easier, users can select which information they want to edit (for example, morphology) and a separate tool is opened to edit just the part of entry structure (see Figure 7.22). The application handles team cooperation and ensures that several users do not overwrite changes of one another.

The PRALED client application allows to edit all the structural elements of a complex dictionary entry:

- part of speech and detailed information about the headword type,
orthography (spelling), hyphenation,
pronunciation,
morphological properties (for the given POS) with the possibility to get these information from the morphological analyzer,
etymology and word origin,
statistical information, automatically inserted from corpora,
linked entries: abbreviations, collocations, phrases, components, synonyms, antonyms, hyponyms,
meaning explanation,
domain, temporal and spatial properties,
examples and corpora concordance.

It is possible to link entries together, for example to refer to dialectic variants, related collocations, phrases or hyponyms. Users can easily open linked entries while editing the entry, they can also select whether to open the entries in the preview or edit mode.

To add a word usage evidence from the corpora, PRALED is connected with the Czech National Corpus [ICNoo]. Linguists are able to select several examples from the corpora and insert them to the edited entry, see Figure 7.23.

For the project managers, daily reports were provided to track the work progress. The reports included filtering by various variables, e.g. entry authors, Part of Speech, letter distribution, or entry status (work in progress, finished...).

Project results
The lexical database was successfully built and finished in 2011. The final database contained more than 1.8GB of data in XML format, and a total of 213,533 entries (for detailed part of
speech statistics, see Table 7.6).

Currently, the lexical database is used in preparation of a new Czech dictionary. The resulting data were included in a dictionary browser used by the lexicographers to gather information, and the users are able to see the data in the same interface as other resources.

Table 7.6: PRALED – part of speech statistics

<table>
<thead>
<tr>
<th>PoS</th>
<th>number of entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>83,028</td>
</tr>
<tr>
<td>adjective</td>
<td>48,848</td>
</tr>
<tr>
<td>pronoun</td>
<td>230</td>
</tr>
<tr>
<td>numeral</td>
<td>7,708</td>
</tr>
<tr>
<td>verb</td>
<td>30,130</td>
</tr>
<tr>
<td>adverb</td>
<td>13,442</td>
</tr>
<tr>
<td>preposition</td>
<td>138</td>
</tr>
<tr>
<td>conjunction</td>
<td>153</td>
</tr>
<tr>
<td>particle</td>
<td>194</td>
</tr>
<tr>
<td>interjection</td>
<td>1,728</td>
</tr>
<tr>
<td>idiom</td>
<td>622</td>
</tr>
<tr>
<td>phrasal</td>
<td>9,700</td>
</tr>
<tr>
<td>proper noun</td>
<td>15,459</td>
</tr>
<tr>
<td>abbreviation</td>
<td>206</td>
</tr>
</tbody>
</table>

7.8 **New Encyclopaedia of the Czech Language**

The first edition of Encyclopaedia of the Czech Language [BKNP02] was published in 2002. It is one of the basic reference books for the study of the Czech language and related linguistic disciplines. However, many new concepts and even new research areas have emerged since the publication. It was decided to prepare a completely new edition of the encyclopaedia, rather than just publishing supplements. The new edition will cover current research as well as describing all the concepts of the linguistic studies. The project started in 2011 and is coordinated by the Faculty of Arts, Masaryk University, with the aim to publish finished encyclopaedia at the end of 2015.

It was decided that the new edition will be mainly published as an electronic encyclopaedia, utilizing various advantages of the electronic dictionaries. The printed version will also be published, but rather as a supplement for the online edition. This move to electronic publishing is in line with the recent trends discussed in chapter 2.3. The DEB platform was selected as the dictionary writing system for the preparation of the new edition.

The team consists of more than a hundred authors. The DEB platform support for complex access rights is utilized and all the users are hierarchically organized as entry authors, entry referees, area administrators, editing assistants, and encyclopaedia coordinator with various level of
access to the dictionary data. For example, authors may only edit the entries assigned to them by the area administrator.

Because the source materials of the original edition were available as a set of word processing documents, and mainly because it was not possible to use the online editing tools for some authors, it was decided that in the first stage of the editing, the entries will be written as a word processing documents. To allow the use of the new features for the electronic encyclopaedia, special markup tags were developed for the project, e.g. for various links, or multimedia files. At the first step, documents provided by the authors in several word processing formats are unified by conversion to the Open Document format $[\text{BDE}^+ 11]$. In the next step, documents are converted to an internal XML format. The word processor instructions and special markup tags are converted to semantic tags of the encyclopaedia XML format. More than 900 documents have already been processed and converted to XML format. After the upload and conversion, the documents are stored in the database and edited in the online tools. It is also possible to download the entries for offline editing and upload the updated version later.

The new edition is accompanied by several features of electronic and multimedia dictionaries to help users with navigation in the encyclopaedia, gather more information, and better understand the concepts. The DEB platform tools take care of properly encoding and providing all the following information:

- cross-references to other entries or entry parts,
- links to external websites,
- references to the bibliography, with the possibility to query external library resources,
- images, charts, diagrams etc.,
- sound recordings (e.g. samples of various dialects),
- animations and video recordings (e.g. video showing sign language).

To make the encyclopaedia comprehensible and more useful for different groups of readers, the entries are described in two different levels of proficiency – elementary and advanced. As of January 2015, out of the total 953 entries currently stored in the encyclopaedia, 662 entries contain only the elementary description, 154 entries contain only the advanced description, and 137 entries have both descriptions.

Readers on the encyclopaedia website may choose their preference of the description level. For example, readers may hide advanced information and when they search for an entry, only the elementary entries or description are provided.

In the final year of the project, all the entries will be globally reviewed, checked for inconsistencies and prepared for final publication. Using the options for different output versions, the source materials for the printed version will be produced from the same database as the electronic online version. For the print edition, advanced features need to be left out (e.g. multimedia recordings) or encoded in different format (e.g. cross-references in different typeface).
Although the projects presented in the previous sections are examples of complex dictionaries or large research project, there is also a demand for dictionary writing software to create and publish dictionaries with more plain entry structure. Several such dictionaries were created using the DEB platform tools. For example, the Terminological Dictionary of Fine Arts by the Faculty of Fine Arts, Brno University of Technology [HR07a], or the Czech-English Dictionary of Ethnological Terminology by the The National Institute of Folk Culture [19]. To fulfill the requirements for such range of dictionaries, a new application of the DEB platform was developed, called DEBWrite. The paper describing the DEBWrite application is to be published in the proceedings of the eLex 2015 conference [Ram15].

The application is implemented as a multi-platform web application, utilizing standard HTML5 and JavaScript, with jQuery [jQu15a] and jQuery UI [jQu15b] libraries. The DEBWrite application allows users to create and share a new dictionary without any difficult configuration or advanced technical skills. Based on experience with dictionaries in the DEB platform, a default entry structure is proposed that fits many dictionaries (also with terminological dictionaries in mind). Each entry is composed of top level information (headword and its variants, grammatical information, domain/category) and any number of meanings (each containing explanation and usage examples). Translations to various languages, cross-references to other entries (with relation type), collocations, and external references may be included on the entry level or meaning level. On dictionary creation, users may alter the entry structure in a graphical interface – deleting unnecessary information or adding new entry fields, changing labels, or altering the option lists (relation types, languages for translations, domains...). If the dictionary authors want to enable customization for the readers, they may set the proficiency level (basic or advanced) for each part of an entry. When browsing the dictionary, readers may display only the information they need for better user experience.

According to the updated entry structure, editing form and the public browser are generated automatically. If users wish to change the website design, they may provide custom CSS stylesheets or XSLT template that are used for output generation. Currently, users need to change the source code of the CSS or XSLT files. In future versions, a graphical interface to change the output layout will be added.

It is possible to import the list of headwords and empty entries are created according to the list. The application also supports import of the dictionary data in XML format, and export to XML, HTML or PDF format.

The application supports three levels of user roles: manager, editor, and reader.

- The user who created the dictionary is the manager. Managers may alter the dictionary
setting. They may grant access to the dictionary to other users, specifying their role. They are able to edit all the dictionary entries and set an entry for publication. The manager may also decide to make published entries publicly available, which means that no password is needed to browse the dictionary.

- Editor may edit the entries that are not published yet.
- Readers may only browse the published entries.

The DEBWrite prototype is currently in testing, with a plan to release it publicly as a free dictionary writing system.

7.10 Chapter conclusion

This chapter demonstrated adaptability and variability of the DEB platform for various research and lexicographic tasks with multi-disciplinary overlaps – e.g. filtering and combining big data resources, heavy multimedia use for the dictionary of sign language, or supporting multi-lingual knowledge detection system. Presented applications are designed and implemented by the thesis author (with co-authors on client software for some projects, referenced in respective sections). All the presented projects have already been applied internationally in various research and study tasks solved by teams in more than 10 countries.
The basic tool for the manipulation of reality is the manipulation of words. If you can control the meaning of words, you can control the people who must use them.

Philip K. Dick

8 Evaluation

Since the release of the first version of the DEB platform in 2006, the number of users and a range of application have been growing. See Figure 8.1 for the user statistic of publicly available resources at the DEB server operated by the Natural Language Processing Centre¹.

Apart from the server hosted by the NLPC, the DEB platform is installed in other institutions. Following list names institutions that either have the DEB server managed by the thesis author, or contacted us regarding the deployment (see Figure 8.2 for the map of installations). Since the complete installation is freely available on the DEB platform website, it may be deployed even more widely.

- Universiteit van Amsterdam – Cornetto project (see chapter 7.5),
- Vrije Universiteit Amsterdam – Cornetto, KYOTO (see chapter 7.6), and DEBVisDic (for Open Dutch Wordnet),
- Institute of the Czech Language AS CR – PRALED database (see chapter 7.7),
- Adam Mickiewicz University, Poznań – DEBVisDic (for PolNet),
- Research Institute for Linguistics of the Hungarian Academy of Sciences – DEBVisDic,
- University of South Africa – DEBVisDic (for South African languages),
- North-West University, South Africa – DEBVisDic (for South African languages),
- University of Ljubljana – DEBVisDic (for sloWNet),
- Teiresías Centre, Masaryk University – Czech Sign Language dictionary (see chapter 7.3)

¹http://deb.fi.muni.cz
Figure 8.1: Number of users that have signed the Licence of the NLPC DEB services at the Brno DEB server

- University of the West of England – Family Names in Britain and Ireland (see chapter 7.1)
- University of Wolverhampton – Pattern Dictionary of English Verbs [MBBH14],
- Fundação Getulio Vargas, Brazil – DEBVisDic (for Brazilian Portuguese wordnet),
- Nanjing Normal University, China – DEBVisDic.

The DEBVisDic application has been used to create and edit more than 13 national wordnets (that we are aware of), for details see chapter 6.3. The DEB platform was also selected as the best option for multiple successful research projects, like Cornetto [HVR08a] or KYOTO [HR09]. Thanks to the implementation of the DEB platform, large lexicographic or encyclopaedic projects, such as Family Names in Britain and Ireland, benefit from incorporation of the wide range of resources in one place. This also leads to faster completion and publication time [HCM11]. It would be much more difficult to compile a dictionary with different editing application.

8.1 Database storing optimizations

During the development of the DEB platform, the database responses were getting slower on large lexicons of complex structured data. Several optimization techniques were evaluated to speed up the database queries. Based on the evaluation, optimizations were implemented for repeated database queries on linked data in the lexicon (e.g. when the entries are linked together by the identifier and the editor need to display data from both entries), effectively reducing processing time from tens of seconds (in extreme cases) to less than a second [HVR08a].

http://www(pdev.org.uk/)
However, this optimization did not solve all the issues. It was decided to evaluate the performance of the backend database engine and compare several available XML database engines. Figure 8.3 presents the results of one of the performance benchmarks. It is apparent, that the Berkeley DB XML used initially as the DEB platform database engine is outperformed by other databases. Thanks to the modular architecture of the DEB platform, it was possible to smoothly exchange the database backend. Based on the evaluation, described in more detail in chapter 6.2.3, Sedna XML database was selected which lead to further performance improvement [HR12, BHR10].

8.2 Linked Open Data rating

Linked Data methodology is described in section 5.5 and the DEB platform implementation in section 6.1.13 Berners-Lee later added also the rating system for published data, while expanding the term Linked Data to Linked Open Data – which means Linked Data that are released under
an open licence. This rating system is aimed especially at government agencies to encourage them to publish valuable information. The importance of Linked Open Data is acknowledged for example by the European Union, funding projects like LOD2 (large integrating project to develop tools, standards and management methods for Linked Open Data) or Open Data Portal (catalogue of data available for reuse). The rating system follows these principles:

• 1 star – the data are available on the web in any format, with an open licence.
• 2 stars – the data are published in machine-readable structured format.
• 3 stars – the data use non-proprietary format.
• 4 stars – W3C open standards (RDF and SPARQL) are used to identify objects for linking.
• 5 stars – the data contain links to other resources to give context.

We will now check whether the DEB platform may be used to publish also Linked Open Data:

1. published online with an open licence – this has to be decided by the data authors, but the DEB platform enables releasing data on the web.
2. available as machine-readable structured data – documents in the DEB platform are stored in an XML format which is machine-readable.
3. non-proprietary format – XML is a standardized format.
4. use open standards from W3C (RDF and SPARQL) – XML format itself is the W3C standard, but to conform with this requirement more precisely, documents are converted to RDF format.
5. link to other resources – the DEB platform enables interlinking to other resources.

As demonstrated, the only limitation is the decision of the data authors regarding the licensing. When that is resolved, the DEB platform enables to publish all documents as Linked Open Data.
This work describes the design and development of the lexicographic platform, named Dictionary Editor and Browser (DEB). During the platform design, requirements for different types of knowledge and language resources were considered. These requirements are based on the most used resource types, existing formal models and standards, and processes for resource creation that are described in chapters 2 to 4. Chapter 5 details the common requirements and compares features of widely utilized software systems that are aimed at various types of language resources. Recent research aimed at enhancement of user experience, effective data access, and information sharing via linked data is also analyzed.

The author has designed and implemented the platform that enables creation, management, and sharing of variable language resources (e.g. ontologies, semantic networks, or dictionaries). Chapter 6 compares the features of the DEB platform with previously stated requirements. The DEB platform fulfills the common requirements and introduces enhancements based on recent lexicographic research, mainly in the integration of multiple resources. More than ten applications were developed by the thesis author on top of the DEB platform, and nine projects using the DEB platform are described in chapter 7. Selected applications have demonstrated the variability and adaptability of the system for the resource management. Presented applications include the filtering and combining of multiple big data resources, integration with the corpus for semiautomatic extension of multilingual terminological thesaurus, multimedial dictionary of sign language, or support for complex editorial processes.
Reception of the DEB platform itself and implemented applications in lexicographic and research community is summarized in chapters 6.3 and 8. The DEB platform is recognized as an established software system to build and manage language resources, and is used by at least 13 institutions around the world. The applications based on the DEB platform were important for the successful outcome of various projects, e.g. EU-supported KYOTO project, cooperation with Oxford University Press on publishing the Family Names in Britain and Ireland dictionary, or building the Czech lexical database. The application for semantic network editing (DEBVisDic) is the most widely used software for wordnet building and sharing. The database format introduced by DEBVisDic is also implemented by the authors of other wordnet applications as an interchange format. The DEB platform also promotes sharing and interlinking of information resources, and supports the methodologies for Linked Open Data.

The author has developed a lexicographic software platform with wide range of advanced features, which is freely available and is already used in more than ten international research and commercial projects. The platform is under active development and new features are being added to allow more users to build and share structured language resources.
Bibliography


Peter Mark Roget. Roget’s Thesaurus. 1852.


A.1 Book chapters


A.2 Journal papers


A.3 Peer reviewed conference papers


author's contribution: 100 %


author's contribution: 70 %


author's contribution: 15 %


author's contribution: 50 %


author's contribution: 70 %


author's contribution: 50 %


author's contribution: 70 %


author's contribution: 30 %


author's contribution: 50 %


author's contribution: 60 %

author’s contribution: 70 %

author’s contribution: 60 %

author’s contribution: 80 %

author’s contribution: 80 %

author’s contribution: 30 %

author’s contribution: 40 %

A.4 Other papers

author’s contribution: 30 %

author’s contribution: 60 %


