Conceptual Data Structures (CDS)

Towards an Ontology for Semi-Formal Articulation of Personal Knowledge

Max Völkel, Heiko Haller

FZI, University of Karlsruhe, Germany
{max.voelkel, heiko.haller}@fzi.de,
http://www.fzi.de

Abstract. In an attempt to create a lean vocabulary for incremental recording and step-wise formalisation of personal knowledge, we identified a set of common knowledge structures. These *Conceptual Data Structures* (CDS) were found to be inherent to a variety of different knowledge artefacts ranging from vague paper notes to highly structured documents. CDS is suitable for representing knowledge in various degrees of formalisation a uniform fashion, allowing gradual migration.
CDS serves two purposes: First, as a guideline for future *Personal Knowledge Management* (PKM) tools, providing a set of crucial structural primitives. Second, the RDF-based representation of CDS can serve as a knowledge exchange format. It is capable of representing also vague or even inconsistent knowledge structures between people and PKM tools without unnecessary loss of information.

1 Introduction

Motivation. There is a wealth of methods and tools that facilitate our every-day Personal Knowledge Management (PKM). They range from hand-written paper notes to personal semantic wikis and from mere text processing or spreadsheet applications over special outlining tools to nice and colourful graphical mind-mapping applications. They all have one thing in common: They help the user externalise knowledge in a more or less structured way.

According to Nonaka and Takeuchi [1], externalisation is the articulation of tacit knowledge, that resides in a someone's mind and is often only vague, into explicit knowledge that can be communicated. The act of externalisation is one of the four conversions in their widespread model of the knowledge creating process, and it is the step that must come before any piece of knowledge can be externally stored or even logically processed.

Although explicit, externalised knowledge still varies largely in its degree of formalisation. It can be anything between a loose collection of keywords, a weakly structured text, an informal graph or hypertext up to highly structured knowledge representations and fully formalised ones like an ontology.

Highly structured and formalised information sources are easier to search and process [2] and allow for semantic processing, but require higher efforts to be created in the first place. For many uses however, a weak structure is sufficient (E. g.: “This issue needs to be solved before that one.” or “These topics are somehow related to those.”)

Our guiding use case is *document creation.* Culturally, documents (letters, emails, scientific articles, newspapers) are an established form for communicating knowledge. A document [1] “... is usually intended to communicate or store collections of data”. In practice, few documents are written directly in a text editor in a linear one-pass fashion. Instead, different tools are employed for different degrees and kinds of structures (e. g. note about the idea, outline, argumentative structure, reference data base, pieces of text from another document).

For each range of formalisation, different tools are suitable. But since no single tool is optimal for all stages, external knowledge—especially while being articulated and formalised—often goes a long way, traversing different media and tools while subsequently gaining its final structure. As most tools do not share a common interchange language, contents need to be rewritten or restructured several times during this process. E. g.: New knowledge might come to life during a brainstorming session where it is recorded in keywords and roughly structured on a Mind-Map. Then maybe a report is written according to this map, taking into account other personal notes. Finally, a formatted text is created. A reader is left with linear order and some parts of the hierarchy and has to mentally re-create the inner structure of the document. Exporting a structured data set to another tool often has the same effect: The text is still there, but all (or most) structure is lost and needs to be articulated again every time it is transferred to another tool.

Idea. We have observed, which structures people use in different tools and use cases and found out, that a small set of relations is very common across many kinds of knowledge artefacts. We believe, that the set of “Conceptual Data Structures” (CDS) presented in this paper can

- act as a formalism for recording, managing, and sharing personal knowledge,
- bridge the rather appreciable gap between “no semantics” and “fully formal semantics” in PKM,
- serve as the least common denominator for knowledge exchange between different humans and different tools, and
- encode pre-verbal knowledge (e. g. “this is *nested within* that, but I can’t say why”).

It is our goal, that PKM tools should be able to also capture vague semantics in a way that can be of use and that can be communicated without unnecessary loss of meaning between tools, but more important, between humans.

CDS allows users to build up huge personal “knowledge bases”, consisting of text snippets and references, structured in varying degrees of formality.

[1] `http://en.wikipedia.org/wiki/Document`

CDS offers by its design three parallel ways to work with personal knowledge:

keyword search for item retrieval, using structural and formal knowledge only for ranking,

structure for retrieval by associative browsing as well for composing documents from existing items, and

semantics for reasoning or e. g. stating argumentative structures.

Outline In this paper, we present our design and sources of inspiration (Sec. 3), the CDS vocabulary (Sec. 3.2, usage scenarios (Sec. 4), an implementation (Sec.5) and compare it with related work (Sec. 7).

2 Analysis

CDS was not created with any mathematical or philosophical goals in mind. Instead, existing user interfaces, file formats and ways of structuring knowledge have been examined. CDS is a pragmatic approach, based on structures common to a large variety of contexts and document formats—among others the following:

Text Documents: Text documents have a linear structure. In fact, this linear structure seems to be the most basic structuring primitive used. The first conceptual relation type of CDS is *order*.

Most documents however carry more structure than only sequentially ordered sentences; they are often structured into headlines, chapters, enumerated or bulleted lists. All these structures are hierarchical relations between text fragments. The second relation type of CDS is *hierarchy*.

Paper Notes: Typical note structures on paper are e.g. to-do-lists (sequential), outlines (hierarchical) or less structured item collections. Often, certain items are emphasized or highlighted.

Hypertext: An ubiquitous document format of our time is (X)HTML, which adds hyper-links to the document tree model. A formalisation of these three axes, order, hierarchy and linking is equivalent to the XML document structure. Directed links are another relation type of CDS.

File Systems: All popular file systems use a hierarchy as their basic ordering structure. Most file systems also allow links (shortcuts in Windows-, symbolic and hard links in Unix file systems.)

Folksonomies: Many web sites allow users to *tag* their content. Delicious[2] assigns single-term keywords to bookmarks, flickr[3] labels digital images. Both systems allow users to browse the implicitly created sets of items which share a common tag. “Annotation” is a CDS relation that also allows tagging.

[2] `http://del.icio.us`

[3] `http://www.flickr.com`

3 Design

In this section we present entities and a set of relations between them.

CDS models knowledge at the granularity of text snippets ("items"). We expect the typical item text length to range from a single word up to a paragraph of a few sentences. Longer texts will usually be represented as a collection of related items. CDS allows a user to explicitly state knowledge more subtle Than e. g.: "this is a totally ordered list" or "this is a set without any order)". Instead it allows pair wise ordering: a user can state which other items are "before" or "after".

3.1 CDS Data Model

At its core, CDS has a very simple data model. It consists of:

Address: Items as well as relations between items (i. e. instances of relations) have a globally unique address string. Note that this different from e. g. RDF [3], where only items are addressable.

Item: the smallest addressable unit of information.

Item Text: Items typically carry text. Text can be as short as a label or as long as a paragraph.

Item Type: CDS defines the types *Relation*, *Background* and *URL*. Items of type URL are considered web resources, and can be de-referenced with the HTTP protocol. The type system is extensible—other types can be defined. An item can also have multiple types or no type at all, which is the default case.

Relation Type: CDS comes with a set of built-in in relation types, placed in an inheritance hierarchy. This allows a tool, which does not know how to handle a specific relation type, to fall back to a higher level, using less formal semantics. This is probably the greatest achievement of CDS.

Relation Instances: The intelligence of a CDS model lies in the relations. Relations instances are also items. Each relation instance can go from a number of items to another set of items. Each relation can also have a number of relation types. Details about relation types can be found in the next section.

CDS Model: A set of items, relations and contexts.

Knowledge Exchange. In order to exchange items with other people or tools, we use a concept inspired by *Published Subject Identifiers* from Topic Maps[4]. In CDS, each item may store a number of "merge-with" addresses, representing alternate addresses of the item, that it has been merged with. Retaining these alternate addresses is needed to avoid duplicates when previously merged items are imported again or when a new CDS Model is imported referring to the old addresses.

3.2 CDS Vocabulary

The main part of CDS is comprised of a set of relation types, modelled after the empirical studies of current common data structures to model personal knowledge. We are confident that a small number of relations can be sufficient for modeling personal notes. Some findings even suggest that eight relations types suffice in most cases[5]. We found the following types (c. f. Fig. 1), including the non-typed relation. :

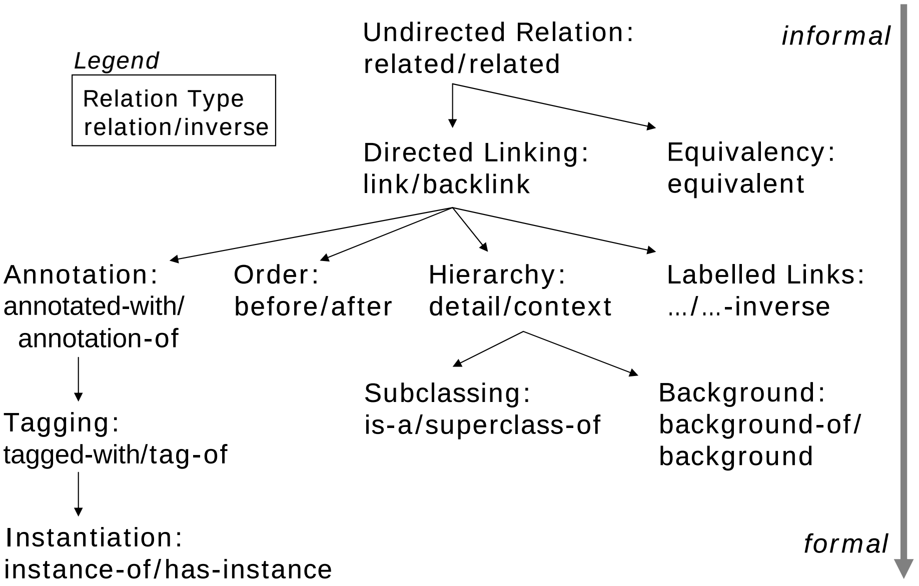

Fig. 1. Inheritance hierarchy of relations

Detail/Context: These two mutually inverse, transitive terms create a hierarchy of items. This hierarchical relation is especially useful for approaches using Levels-Of-Detail (LOD)[6], e. g. like simply collapsing/expanding an item or so-called semantic zooming, where details of an item become visible only when there is enough space to show them.

Order: To enable (partial) order between nodes, we introduce the mutually inverse terms **before** and **after**. Before and after are transitive. They can be used to record any order, be it timely, causal or in degrees of importance.

Directed Links: Web pages and scientific documents often have links between the different content trees. CDS offers **link** and the inverse property **backlink** to create hyperlinks between nodes.

Annotation: An annotation is a simple way of adding a comment to any item. Annotations are frequently used in many tools and contexts ranging from highlighting text passages in a newspaper or sticking post-its to your door to adding text passages to branches of a mind-map or using the comment-function in a text-processor. CDS uses **annotated-with** and the inverse **annotation-of**.

Tagging: A way of annotating that has become quite popular recently is tagging, a tag being formed simply by a short annotation, typically only one word. Tagging creates labelled sets which imply no further information. Tagging is not transitive, which distinguishes it from typing. Tagging is considered the simplest form of assigning meta-data to items. CDS uses **tagged-with** and the inverse **tag-of**.

Typing: Typing (or: instantiating a type) of items is a common concept and can be seen as a more formal way of tagging. It is also sometimes referred-to as *semantic annotation.* Types are often structured in hierarchies and higher types are inherited (type hierarchies are transitive). In CDS the type of a node is stated with **instance-of** and has the inverse relation **has-instance**. Types—like anything—are represented by items. Note that the type system is part of the knowledge structure. Type inheritance is modelled with *Subclassing*, offering the relations *is-a* and its inverse *superclass-of.*

Typing and annotation (tagging) are similar in nature. However, in practice they often serve different purposes: Types dictate how items are processed or rendered while tags help more in finding and grouping items. Typically, items have one or few types and many tags. In CDS, typing an item implies tagging it with the label of the type.

Equivalency: A user may state that two items are *equivalent.* This retains the individual item identity, but searches will treat the two (or more) items as being a single item, having the union of linked items. The benefit of relating items with equivalency instead of merging them, is maintenance: Later, the very same items might be split again. E. g.: A person who is currently, but not forever, my boss. It does not make sense to merge these two items, as this would blur the distinction between these two roles.

Background: Additionally, each item, especially each relation instance can belong to one or more backgrounds. This is stated by a relation typed "background" pointing to any item representing some kind of context. Among other things, this allows to nest backgrounds. Backgrounds are used to record the creator of items and relation as well as the time, when such data has been stated. This information can later be exploited e. g. in ranking a search result.

3.3 CDS with Pen and Paper

As CDS is intended to model private notes, we believe it makes sense to have a graphical notation. Destilling from our own experiences with taking notes on paper and building upon existing notations for personal note taking [4], we came up with the notation depicted in Fig. 2. This is just a first attempt. Ideally, this proposal would eventually evolve into a *Personal Modeling Language.*

[4] http://www.rzuser.uni-heidelberg.de/~x28/anke/text/

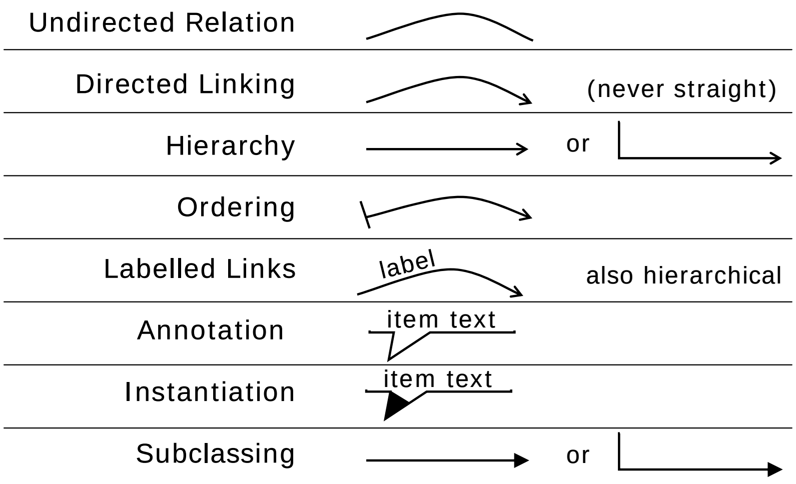

Fig. 2. A graphical pen and paper notation for CDS

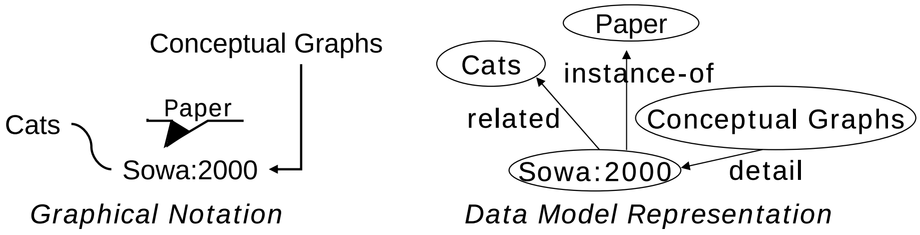

Fig. 3. An example showing the use of the CDS graphical notation (left) and data model (right)

To give a better understanding, how CDS and its graphical notation can be used, we describe the paper 'Sowa:2000' belonging to the field of *Conceptual Graphs* and being somehow related to *cats*(c. f. 3). In CDS, we encode this using four items: *Sowa:2000* (a), *paper* (b), *Conceptual Graphs* (c), and *cats* (d). We could state these relations: `a instance-of b`, `b context c`, and `b related d`.

3.4 Extending CDS

CDS by itself is not very expressive. We expect most users to extend it to suit their needs, by creating new relation types in addition to the pre-defined ones. This is achieved by linking a relation to an item representing a type. Any item can act as a type. New relation types should inherit from existing ones, allowing tools to fall back on their semantics, if the new type cannot be understood. E. g. *metAtAConference* could imply *knows*, which in turn might be layered on *relation*.

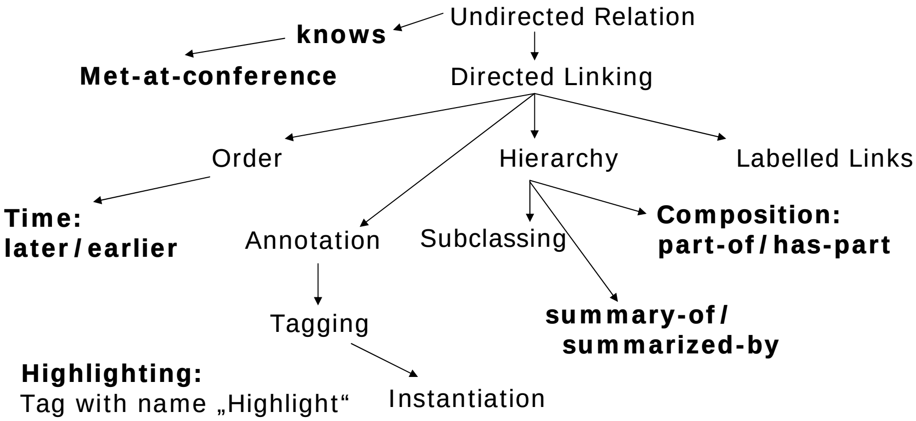

Fig. 4. Extending CDS

Existing ontologies can be used in CDS. However, they should be aligned to the CDS core relations, in order to get the advantages outlined in Sec. 1. An exemplary layering is shown in Fig. 4. Some possible extensions to CDS are:

Vague Knowledge: In order to record vague knowledge as well, CDS could state the **degree** of each relation as a numeric floating point value between 0 and 1. By default, a degree of 1 (100%) would be assumed. On paper, this could map to thicker or thinner lines. Technically, a new "degree" relation would attach items to relation instances. The attached items would than carry the numeric value.

Queries: To fully exploit the structured knowledge represented in CDS, a query language is needed. We believe that SPARQL [7] is a suitable language for that. The formal semantics of the CDS type system would however still need to be clarified.

Aliases: In textual environments, items might need one or more identifying keywords, much like a WikiWord [8]. Such *aliases* should be stored in CDS as well and could so to be shared with others.

4 Using CDS

In practice, at least the novice user is not meant to be bothered with explicitly stating CDS relations. CDS should be stated by the PKM tool, wherever it identifies such underlying structures. E. g. Mind-Maps or outlines *imply* context/detail relationships and orders.

The process of writing a document, as we have it in mind, could be like this: Text snippets are created and structured in a textual interface (like a Semantic Wiki[5]). The structuring could take place by simple wiki syntax (e. g. for nested lists etc.) and a semantically enriched link syntax[9], to explicitly state CDS and other typed relations where desired by the advanced user. But even without these, simple document structures imply many structural decisions: E. g. order and hierarchy of each single line of text have been assigned. As a next step, the CDS data could be refactored and refined in a graphical user interface.

Because it is possible to state local order relations between items, there is no need to overlook the global structure of the document in every phase of its creation. Of course this can produce intransitivities (circles) or other inconsistencies. This is a feature, not a bug: Unfinished, and inconsistent structures can be stated as they occur in transient stages, and shared between tools or people for further refinement. Such inconsistencies could be detected by a good tool and finally resolved by the user.

In our example, a final total order could be determined, (stored by assigning a complete chain of before/after-relations that have a common background item). This background item now defines the structure of the final document, which could now be exported as a stand-alone document or shared as a CDS Model, thereby preserving all the additional fine-grained structure. Ideally there should be new stand-alone (e. g. xhtml-based) document formats, which are highly structured on more fine-grained levels. They could be interactively browsed by the reader rather than linearly read. A first step into this direction is the ABCDE-Format[6].

A good PKM tool should allow the user to start with completely informal text structures and refine his knowledge base by-and-by, adding more specific relation- and item types. In the end, parts of the personal knowledge base could be as formal as an ontology.

5 Implementation

CDS is currently implemented as an RDF Schema using some OWL fragments. We sketch the implementation only briefly, the complete RDF Schema can be downloaded from `http://xam.de/2006/01-cds`. Embedding CDS in RDF allows an easy route to transform vague knowledge step-by-step into hard factual knowledge. RDF is well known for its ability to encode complex, formal knowledge structures. The CDS vocabulary can model vague knowledge as well.

Items are instances of the class `cds:Item`. The textual value is assigned with `rdf:value`. Different from RDF, CDS needs to address relation instances as well. We thus model all relation instances as relations of the class `cds:Relation`, following pattern 2 from [10]. Each instance of a relation has a number of *source* items, *target* items and items representing the *types* of a relation. We now give an illustrative example for encoding CDS information (as stated in Fig. 3) in RDF, which is depicted in Fig. 5.

6 Evaluation

In this section we evaluate the ability of CDS to model basic structures of existing PKM tools. We conclude this section by listing current shortcomings.

Use Case: Wikis How typical structures occurring in a wiki (links, backlinks, hierarchies from headings and nested lists etc.) can be dealt with, has allready been described in Sec. 4.

Use Case: Mind-Maps [11] As mentioned above, the basic hierarchical structure of mind-maps corresponds directly to CDS' context/detail relation. The sequential order of branches can be described by before/after. Cross-links in mind-maps can be expressed by directed links—labelled or not.

Use Case: Concept Maps [12] Concept maps are graphs without formal semantics, simply using labelled nodes and labelled directed links. The mapping to CDS is obvious.

[5] `http://en.wikipedia.org/wiki/Semantic_Wiki`

[6] `http://wiki.ontoworld.org/index.php/ABCDEF`

```
a cds:label ”x”.                      f cds:from c.
b cds:label ”paper”.                  f cds:to a.
c cds:label ”Conceptual Graphs”.      f cds:instance-of cds:detail.
d cds:label ”cats”.                   g cds:from a.
e cds:from a.                         g cds:to d.
e cds:to b.                           g cds:instance-of cds:rel-relation.
e cds:instance-of cds:rel-type.
```

Example in N3 notation. a-g are URIs.

Fig. 5. Mapping CDS to RDF

Use Case: TheBrain [7] TheBrain is a PKM tool known for its nice and fluent visualisation of local graph structures. It uses two relation types: hierarchy and undirected links. Unlike many other graph-based tools, it allows to model circular intransitivities with the hierarchy relation. As we know, all of this is easily done in CDS.

Use Case: Notes on Paper According to a methodology of Matthias Melcher [8], there are a number of fundamental relations for personal knowledge items. We explain how they are expressed in CDS.

Part-whole can be seen as a refinements of context/detail.
Tags and taxonomies map to CDS' ability to use tags and types.
Order, either causal or in time both maps to before/after.
Synonyms can be stated with "equivalent".
Cross-Links are modelled with CDS' general relation.
Antonyms have currently no built-in way to be expressed in CDS. Maybe the ad-hoc introduction of a new directed relation would be the best option.
Marking items is offered by tags.
Citations and references map nicely to hyperlinks.

6.1 Shortcomings of CDS

Tables are not natively handled in CDS. Simple tables often represent a list of objects with a list of attributes each, which could be modelled that way. However handling the semantics of tables generically is not trivial [13]. Especially the handling of nested tables is currently unclear.

Numeric knowledge or even charts are currently out of scope.

7 Related Work

In this section, we briefly review existing knowledge representation languages and vocabularies.

RDF is rather technical and not ideally suited for direct conceptual modeling. RDF focuses on merging models, where items having the same URI are considered equal. In order to use RDF in a scenario, some kind of vocabulary or ontology is needed, CDS tries to be this generic PKM vocabulary.
OWL can be considered in the context of this papers as a mighty type system. OWL does not directly allow to use as unprecise relations as *more general than.* However, expressing CDS in OWL seems possible and will be considered in future versions.

[7] `http://www.thebrain.com/`
[8] `http://www.rzuser.uni-heidelberg.de/~x28/en/102.htm`

Topic Maps [4] tackle mostly the problem of creating digital indexes over heterogenous items (physical items, physical information resources, online items, online information resources). There are no predefined relations between items on a semantic level. However, some topic map based tools define foundational relations: *DeepaMehta* [14] e. g. mostly relies on UML relations.

Conceptual Graphs (CG) [15] are a generic tool to record formal knowledge. They have a human-readable graphical notation and a formal interpretation. Like RDF, Topic Maps or OWL, they are completely generic, leaving the user with much freedom. Note that CG are more elaborate about context handling than RDF and OWL. The expressivity of CG is roughly comparable to RDF, but RDF has the greater tool support.

PIMO [16] the *Personal Information Management Ontology*, is quite similar to CDS. CDS, however, is unique in its richness of predefined relations for structuring personal knowledge. PIMO and CDS both emphasise the importance of inverse relations.

IBIS stands for *Issue Based Information Systems* and stems from the background of supporting political decision processes [17]. The central concepts in IBIS are *question*, *idea*, and pro and contra *arguments*. An RDF vocabulary for IBIS has been published [9]. IBIS is structuring thoughts only as items of a discussion, not in the general sense. A graphical tool for using IBIS, gIBIS, has been described in [18].

SKOS is the *Simple Knowledge Organisation System*. The goal is described as "providing a simple yet powerful framework for expressing knowledge organisation systems in a machine-understandable way."[10] It has been developed to express shared, multi-viewpoint, multi-lingual concept hierarchies. SKOS shares some concepts with CDS, but is more technical. CDS is simpler and can describe knowledge more fine-granular than SKOS. E. g. in SKOS one does not define partial order in a set; a set is either ordered (OrderedCollection) or not (Collection). SKOS has means for different kinds of labels and notes for a concept, where CDS allows only one string per concept. CDS is intended for personal usage. The CDS implementation contains SKOS mappings, as far as possible.

Existing knowledge representation languages are very general (RDF, RDFS, Topic Maps, OWL, CG) and come with few semantic relations to capture and structure personal knowledge. They require a quite technical mindset, e. g. expressing all relations as binary relations (RDF, OWL) and thinking about the distinction between literals, blank nodes and URIs. Existing vocabularies and ontologies (SKOS, IBIS) are too domain-specific to model arbitrary personal knowledge.

CDS should be *easier to use than e. g. RDF or Topic Maps*, as it offers a basic set of structuring primitives and thereby guides the user.

[9] `http://dannyayers.com/xmlns/ibis/`

[10] `http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20050510/`

8 Conclusions and Outlook

CDS serves two purposes. First, it acts as a guideline for *Personal Knowledge Management* (PKM) tools. Each tool should be able to create, represent and manipulate at least the concepts defined in CDS. Second, the RDF-based mapping of CDS can serve as an exchange format between PKM tools of all kinds.

We plan to integrate CDS support into a visual mapping tools (iMapping[11]) and a semantic wiki. An alignment with existing ontologies and user studies are planned.

Acknowledgments: Research reported in this paper has been partially financed by the EU in the IST-2003-507482 project Knowledge Web (see [12]), SEKT under contract IST-2003-506826 and NEPOMUK (IST 027705). Many thanks to Benjamin Heitmann, Felix Kugel, Matthias Melcher, Pär Lannerö, Kristina Groth, Jörg Richter, Moritz Stefaner, Thomas Herr, Ingo Frost, Jens Wissmann, Karsten Trint, Daniela Kalb, and Abiba Saibou for having never ending discussions with us.

References

1. Nonaka, I., Takeuchi, H.: The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation. Oxford University Press (1995) Cited in Sec. 1.
2. Staab, S., Studer, R.: Handbook on Ontologies (International Handbooks on Information Systems). Springer (2004) Cited in Sec. 1.
3. Hayes, P.: Rdf semantics. Recommendation, W3C (2004) Cited in Sec. 3.1.
4. Durusau, P., Newcomb, S.: Topic maps reference model. Iso committee draft, ISO 13250: Topic Maps (2005) Cited in Sec. 3.1 and 7.
5. O'Donnell, A.M., Dansereau, D.F., Hall, R.H.: Knowledge maps as scaffolds for cognitive processing. Educational Psychology Review **14** (2002) 71–86 Cited in Sec. 3.2.
6. Furnas, G.W.: Generalized fisheye views. In: Human Factors in Computing Systems CHI '86. (1986) 16–23 Cited in Sec. 3.2.
7. Prud'Hommeaux, E., Seaborne, A.: Sparql. W3C TR working draft (2005) Cited in Sec. 3.4.
8. Leuf, B., Cunningham, W.: The wiki way: Quick collaboration on the web. Addison-Wesley (2001) Cited in Sec. 3.4.
9. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic wikipedia. In: Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, May 23-26, 2006. (2006) Cited in Sec. 4.
10. Noy, N., Rector, A.: Defining n-ary relations on the semantic web: Use with individuals. Technical report, W3C Working Draft (2004) Cited in Sec. 5.
11. Buzan, T., Buzan, B.: The Mind Map Book: Radiant Thinking - Major Evolution in Human Thought. BBC Active (2003) Cited in Sec. 6.

[11] `http://wiki.ontoworld.org/wiki/iMapping`

[12] `http://knowledgeweb.semanticweb.org`

12. Novak, J.D., Gowin, D.B.: Learning how to learn. Cambridge University Press, New York (1984) For a crisp overview on “The Theory Underlying Concept Maps and How To Construct Them” by J.D. Novak, see `http://cmap.coginst.uwf.edu/info/`. Cited in Sec. 6.
13. Pivk, A., Cimiano, P., Sure, Y.: From tables to frames. Elsevier’s Journal of Web Semantics: Science, Services and Agents on the World Wide Web **3** (2005) 132–146 Selected Papers from the International Semantic Web Conference (ISWC) 2004, Hiroshima, Japan, 07-11 November 2004. Cited in Sec. 6.1.
14. Richter, J., Völkel, M., Haller, H.: Deepamehta - a semantic desktop. In Decker, S., Park, J., Quan, D., Sauermann, L., eds.: Proceedings of the 1st Workshop on The Semantic Desktop. 4th International Semantic Web Conference (Galway, Ireland). (2005) Cited in Sec. 7.
15. Sowa, J.F.: Conceptual graphs for a data base interface. IBM Journal of Research and Development **20** (1976) 336–357 Cited in Sec. 7.
16. Sauermann, L.: PIMO-a PIM ontology for the semantic desktop. Technical report, DFKI GmbH, Kaiserslautern (2006) 15.2.2006 version, draft. Cited in Sec. 7.
17. Kunz, W., Rittel, H.W.J.: Issues as elements of information systems. Technical report wp-131, University of California, Berkeley (1970) Cited in Sec. 7.
18. Conklin, J., Selvin, A., Shum, S.B., Sierhuis, M.: Facilitated hypertext for collective sensemaking: 15 years on from gIBIS. In: 8th International Working Conference on the Language-Action Perspective on Communication Modelling (LAP 2003). (2003) Cited in Sec. 7.