Semantic Competence Modelling – Observations from a Hands-on Study with HyperCMP Knowledge Graphs and Implications for Modelling Strategies and Semantic Editors

Matthias Patrick Dahlmeyer[1]

Abstract: Following previous postulations for a global, integrated competence management system, this paper publishes and evaluates reflections from a 2019 study of hands-on semantic competence modelling. Mechanical engineering bachelor and master students explore modelling their technical domain as a study project. They use a previously derived knowledge hypergraph structure (branded herein as HyperCMP) to model key subdomains in a two-staged process. Modelling subdomain knowledge as the first stage was prioritized. In the end, deriving actual competencies had to be suspended because of the lack of a suitable modelling tool that allows managing the complexity of such a model. Observations and reflections from tool research and the modelling process are used to narrow down the profile of a proposed semantic software solution to build, maintain, and use a decentralized competence model.

Keywords: HyperCMP, competence modelling, ontology editor, graph database, reification, hypergraph, federated knowledge graph, visualization, mechanical engineering.

1 Introduction

In previous publications, the strategic need was presented to define and establish a global, decentralized digital infrastructure for an integrative competence management system ([Da06], [DSR16], [Da19]), allowing participative management of life-long learning from individual personal, educational, and organizational perspectives. The underlying functional framework features a distributed, pervasive, and digitally operable competence representation model as a key element. In [Da19], the author derived guidelines for the core model's type and structure from an analysis of functional requirements.

As a follow-up in this workshop, hands-on experiences from modelling competencies in a technology-based domain are presented and evaluated as a set of two papers: The first paper [Da20] portrays more in detail the practical problems of a taxonomic modelling approach, and why this is unapt for decentralized competence management. The second (this) paper explores the process and challenges of a semantic approach, including the search of semantic tools for non-computer experts to model domains as ontologies or graphs, and their implications for semantic modelling strategies and tool development.

[1] Hochschule für Technik und Wirtschaft Berlin – University of Applied Sciences, Faculty 2: School of Engineering – Technology and Life, Wilhelminenhofstraße 75A, Berlin, D-12459, Matthias.Dahlmeyer@htw-berlin.de

10.18420/delfi2020-ws-109

2 Underlying concepts: Competencies and HyperCMP meta-model

The following section compiles the principle concepts for the further paper.

Based on [Da19], *competence* will herein refer to *a semantic network of representations of a perceived reality, dispositioning its owner to cause an effect or induce a change within himself, or to someone / something without himself.*

Semantic modelling of competencies is often understood as a process, where competencies are formulated as a monolithic concept label or *vertex*, and then enriched by subsequent semantic interrelation statements or *edges* (typically called *node* resp. *relations* in ontologies, but furtherly, graph terms will be preferred). Labels can be derived from controlled vocabularies or taxonomies that may already include limited, usually mono-hierarchic semantics. An example was presented by the OpenVM project [BK19], using a semantic extension for competence descriptors in Mozilla Open Badges, based on the ESCO taxonomy [EC20]. This approach is low-threshold for competence formulation – But for a meaningful semantic model, each label will have to be semantically mapped to all other label variations. In a decentrally managed model without controlled vocabulary, this entails extensive effort for modelling, maintenance, and working with the data, even for small networks (see Fig. 3 top left). Also edge combination may be correct but ambiguous (highlighted gray: soup *is a type of* food; salted *is a type of* spiced), or contradicting (highlighted black: potato soup *is a type of* soup; salted *is a type of* spiced).

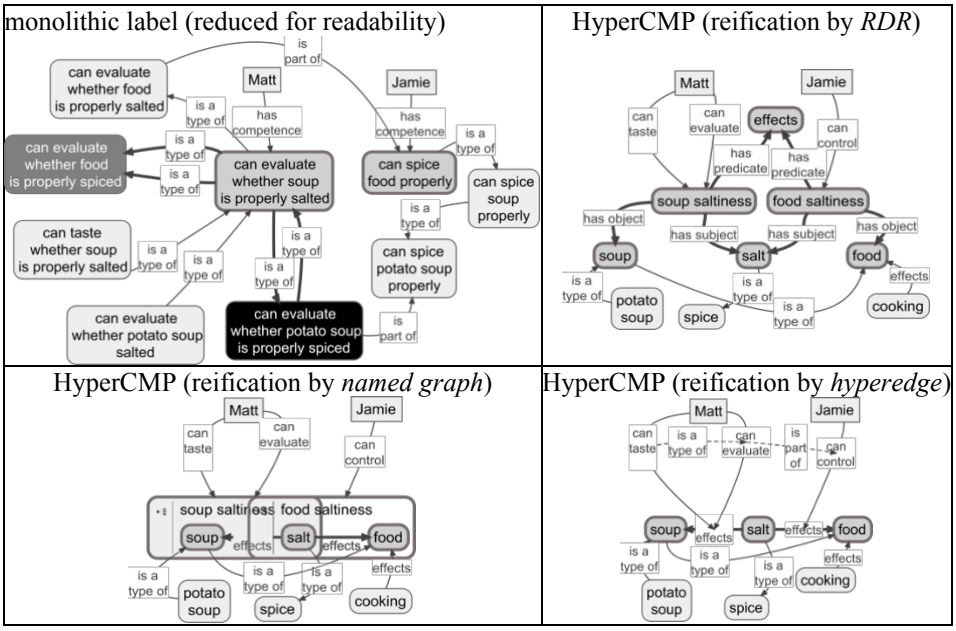

Fig. 3: Comparison of various modelling approaches

As an alternative, a modular semantic modelling approach has been proposed that allows composing competencies from non-controlled elements, but in a shared syntax of *quadruple statements* [Da19]. This approach defines a composition pattern (syntax) for competencies, furtherly referred to as *Hypergraph-based Competence Modelling Pattern (HyperCMP)*: Semantic triples ("[subject] – [predicate] – [object]") are used to lexically[2] model elements of a domain and their semantic structure (what to be competent *about*). Competencies would then be compositionally[3] modelled by addressing these lexical triple statements with a fourth element, a competence owner's *disposition* to effect the respective aspect of the domain (what to be competent *for*, *disposition to* <*x*> [Da19], or *operation verb* [Da20]). This shared syntax allows semantic structures to be distributed over a network of federated domain knowledge graphs while a shared syntactic logic allows cross-linking and cross-interpretation. It also allows to control traversal of a semantic graph *per element* ("semantic search" as described in [Da19]), so vertices and edges can be identified or matched from semantically related (e.g. *synonymous*, *translated*, *broader*, or *narrower*) keywords – including the semantic structure of edges themselves. This reduces model and modelling complexity significantly (see Fig. 3, bottom right), while allowing more specific control *per element* over the semantic search. New competencies can be composed based on previously modelled domain knowledge graphs.

When a fourth semantic component (in this case disposition) points to a semantic statement (*statement about a statement*), this is generally referred to as *reification*. Reification requires quadruples instead of triples, e.g. "*[can operate]* – [hammer] – [drives] – [nail]". In a graph, this is realized by h*yperedges* (edges that do not point to a vertex, but to another edge that represents the triple statement). A graph natively supporting hyperedges is referred to as a *hypergraph* (see Fig. 3, bottom right). Another way to implement quadruples is a named graph [Ca05], with an auxiliary entity for the statement to be reified, e.g. *[can explain]* – [nailing]; with: [nailing] = [hammer] – [drives] – [nail]). Fig. 3, bottom left, displays two named graphs *soup saltines* and *food saltines*. In triple-based semantics systems, reification has to be modelled as an indirect construct, established as *Reification Done Right (RDR)* [HT14]: The statement to be reified is sub-modelled as a vertex which is the object of the reifying statement, but also the subject of three more semantic statements (*has subject*, *has predicate*, *has object*) (see Fig. 3, top right). Such substitutions for every statement of a graph are cumbersome to model and query/traverse (all HyperCMP examples in Fig. 3 contain only two reified statements, to be extrapolated for more complex graphs). The syntax for the example of nailing demonstrates the complexity of this approach: "[person] – [can operate] – [nailing]; with: [nailing] – [has subject] – [hammer]; [nailing] – [has predicate] – [drives]; [nailing] – [has object] – [nail]".

HyperCMP using hypergraphs was premise for observations and implications throughout the study and especially in the tool research – even though it could not be implemented.

[2] as introduced in [Da20]: interrelating semantic elements to define their meaning.
[3] as introduced in [Da20]: composing competence formulations from semantic elements.

3 Methodology

The goal of the study was to explore the modelling process with HyperCMP with open outcome and to understand implications for modelling strategies and semantic editors. Therefore, this paper compiles and evaluates qualitative, subjective testimonies and emerging reflections for what worked and what did not.

The study was conducted in an interdisciplinary project-based course in a mechanical engineering program at the Hochschule für Technik und Wirtschaft Berlin – University of Applied Sciences. Criterion for participation was only to opt for the specific, untypical project task over other more typical engineering tasks.

In October 2019, one bachelor and one master student volunteered for a study project to start an ontology- or graph-based competence model for the mechanical engineering domain. The results were intended to be publically available as linked open data, to support learning by exploration of the domain itself and, later, of mapped content, and for further exploratory research of competence modelling.

Initially as a side-track, semantic modelling tools were researched for utilization in the study. This evolved into a substantial immersion – The results are documented in section 4. The project task for each participant was to choose, research, and model competencies from one key subdomain. In a first step, participants should lexically model their subdomain's content, and in a second step, compositionally model competence statements (about the lexical model) with disposition hyperedges. Participants were also asked to reflect and document their own modelling experiences. The observations and reflections are excerpts from their project reports [Bu19] and [Sc19] (unless marked otherwise, e.g. comments in curly brackets). The project course was held and documented in German. Critical data will be translated or summarized selectively.

4 Research of tools for semantic modelling of competencies

In order to provide a modelling tool for the study, semantic tools were researched for their suitability to HyperCMP modelling. Initial requirements were:

- a system to create a semantic structure like an ontology or knowledge graph (focused on vertex and edge entities only, ignoring semantic attributes).
- natively supporting reification resp. hyperedges.
- browsing and editing appropriate for non-computer experts, i.e. no coding in application programming interface (API) or query language (QL) is required.
- intuitive graphic user interface for front-end tool to navigate and explore data ("browser") as well as to create and maintain data ("editor").
- data can be edited online or updated to a data server for machine-readable traversing and linking into the data. Ideally, the system itself offers web-based

browsing and editing, with differentiated access rights management (ARM). Alternatively, a separate compatible web-based editor is available.
- Availability in descending priority: open source, free, or low-cost academic license for research.

No system was found to fulfill the requirements without major compromise. To put this into perspective, the search was a side track out of necessity and, to a notable degree, new territory of expertise. Some systems were installed and tested, others were abandoned because of the web documentation. Unfortunately, no systematic documentation and evidence per system and aspect was systematically planned – The search began as a simple web search and was refined more and more based on findings. However, within these limitations, relevant insights crystalized in the process, to be shared as overall evaluation of software categories regarding need for action.

A key insight was to distinguish between semantic functionality of data storage, schema browsing, data browsing, data layout & presentation, schema editing, and data editing.

Ontology editors and graph databases often discriminate two different layers: the model's schema layer[4] and the model's data layer[5]. In ontology editors like Protegé, data is strictly instantiated from a schema of binding class taxonomy. A graph can be explicit (strictly binding), hybrid (partially binding) or implicit (not binding). Even though the schema and the data layer basically consist of vertices and edges, visualizing and editing is usually implemented as separate tools or functions.

Ontology editors seem to be ideal for use cases when the schema layer is centrally controlled by few experts and shared, to be populated with data in a close decentral but coordinated cooperation. E.g., Protegé (Desktop) includes an editor for the complex schema, and a more basic interface to create and edit data (as well as multiple graphic data exploration interfaces not capable of editing). Functionality of a local desktop installation can be extended by a large choice of plugins (for WebProtegé very limited). The WebVOWL editor (Protegé plugin or webclient) allows highly comfortable browsing and editing – but exclusively for the schema layer (no visualization and editing of the data layer). All these characteristics of ontologies do not make this software class suitable as a vehicle for HyperCMP modelling: Most of the schema is based on non-taxonomic, subjectively controlled elements, restricted only by syntax conventions.

Graph databases like AllegroGraph, ArangoDB, Blazegraph, Caley, Grakn, GraphDB, GraphEngine (formerly Trinity), HypergraphDB, or JanusGraph (formerly Titan), or Neo4J are apparently preferred for storage of large data sets for analysis or for ad hoc processing. They are notably often available as a mere server, with a text console and application programming interface (API) instead of a graphical user interface (GUI), with few exceptions described below. Many tutorials focus on coding the schema or loading data into the graph in query language, or how to implement queries by software clients.

[4] vertex and edge *types* in a *graph schema; concepts* or *classes* in an ontology
[5] specific *vertices* and *edges* in the *graph data*; actual *instances* or *individuals* in an ontology

Some systems include a visualization window or separate tool, parallel to editing via console (like Neo4J Bloom, Browser for JanusGraph, or the now web-based Gruff for AllegroGraph, which is also promoted as a "graphical query builder").

But a key problem is that most ontology editors and graph databases are not promoted to natively support reification / hypergraphs. Named graphs as a reification concept have been published 15 years ago [Ca05], and they do form part of the SPARQL Protocol and RDF Query Language specification. However, non-IT domain experts cannot be expected to code their knowledge into RDF editors.

However, there are some exceptions: HypergraphDB, Graphbrain, GraphEngine, and Grakn claim to natively support hypergraphs, with Graphbrain explicitly promoted as experimental (version 0.3.2, command-line interface, and MacOS only). But none could be confirmed in the study to have a GUI function or extension for graph building and populating and were therefore not or only superficially installed and explored with the exception of Grakn: Grakn Labs promotes heavily native hypergraph support as a key feature of the Grakn Core database and offers Grakn Workbase as an unpretentious but graphic, force-directed editing tool. In his just published extensive overview of emerging semantic technology, [Fi20] comes to the conclusion that hypergraph are supported by "some of the emerging databases", but afterwards names only that he identified Grakn when writing this article. However, currently in Grakn, only the schema layer can be graphically edited (with some teething problems that have yet to be smoothened out). Unfortunately, competence modelling with the HyperCMP syntax is more concerned with the data layer than the schema layer, and the GUI for the data layer currently only allows browsing – Without graphic data editing functionality, domain experts would have to enter their model using GraknLabs query language GraQL via a text console. Another obstacle for research and distributed modelling is that even simple access rights management (e.g. to prevent editing from website visitors) currently requires the paid Enterprise suite (The basic software is free and open source). Grakn uses an own, yet open source query language for its functionality.

However, during research for this paper (after the study), AllegroGraph (Frank Inc.) was discovered to support hypergraphs by means of named graphs, and it includes Gruff as a free additional tool for advanced graphic visualization, query building, and – actually – editing. This seems to be a promising starting point for further exploration.

Many other graph databases require third-party (although often free) tools for browsing, analysis, and layout, like Cytoscape, Gephi, Graphileon, Graphviz, or VisGraph. None could be confirmed to provide editing the source graph data graphically, or to support reification with again a few exceptions described below. A few tools arrange data in mere fixed planar pan and zoom layouts, ill-suited for large networks. But most visualization tools offer – at least as a choice – a comfortable force directed (or *spring*) layout based on a *Model-View-Controller (MVC)* concept. The prevalence of the force-directed layout method is also attested by the extensive overview and analysis of visualization tools and methods in [Du18] (although no proof was seen that it is significantly better than other

visualization methods). Some tools allow also defining "perspectives" which simplify (filter) the data graph for a specific exploration interest. This could support visual editing significantly.

Franz Inc.'s Gruff is promoted to browse and edit data with named graphs (to be tested in detail). Hyperscape was discovered while writing this paper, with the claims to not only visualize but also edit hypergraphs (to be tested for the schema and for the data).

Since no ontology or graph editor could be confirmed during the project to match the requirements, a basic graph-drawing editor was chosen for the modelling. From two identified freely available tools, the participants chose *Visual Understanding Environment (VUE)* from Tufts University (open source, based on the VUE.js framework) over *yEd Graph Editor* (yWorks, free): VUE's interface is optimized for quick-and-simple editing and basic analysis, rather than yEd's powerful layout and presentation options. Also, resulting data was considered more future-proof for extension because of native hyperedge-support and RDF-export as options for subsequent modelling.

5 Observations and reflections from a modelling study

The bachelor student chose to model *production technology*, the master student *engineering design*. To prevent duplicates, they were to define interface points for cross-reference between the two subdomains. It was decided to prioritize modelling the domain content, and consider the hypergraph-based modelling of actual competencies as a subsequent option.

edge	type	implied reverse	reverse type
[is a type of]	hyponym to	[comprises]	hypernym to
[is the same as]	synonym to	[is the same as]	synonym to
[is part/member of]	partitive/meronym to	[consists of/contains]	partitive/holonym to
[is cause of]	causative cause	[is effect of]	causative effect

Tab. 1: semantic edge type used in the study

A basic set of four edge types was predefined (see Tab. 1) to ensure compatibility of sub-domains but to be extended if required. Individuals of edges had to be type-aligned manually because it was not recognized during the study that VUE – other than most graph drawing tools – does actually support a systematic type definition. Reverse edge types were not explicitly modelled but can be semantically reasoned (e.g. "engine – is part of – car" ➔ "car – contains – engine"). Since [Bu19] considered the basic set insufficient, he introduced additional edge types in his sub-domain of production technology ("A defines B", "A develops B", "A influences B", "B depends on A"). Those can be interpreted as more specific causative types ("depends on" is reverse). Therefore, subdomains are compatible as long as semantic aggregation of edge types is available. [Bu19] recognizes that choice of edge types may be subjective, depending on context. This observation

indicates that edge types should also be subject of decentralized, federated semantic modelling and interpretation, rather than be strictly centrally prescribed.

The students assembled concepts for vertices from technical books, lecture notes, classifications from technical standards, technical websites, and lecture notes, and then sorted, specified, and interrelated them [Bu19].

Lexical subdomain modelling with a graph-drawing tool turned out to be a major challenge in itself. The lexical models quickly grew too complex to manage with a simple drawing tool with planar layout (see Fig. 4 for an impression of result complexity). A full compositional HyperCMP model with hyperedges for competencies would have added significantly more complexity and had to be suspended for subsequent research. Nevertheless, the explored modelling principles are highly relevant also for the subsequent compositional competence modelling.

Some basic cross-subdomain-references were created, including interface points for prospective subdomains (engineering mechanics, quality management) but only highlighted manually without function at this stage.

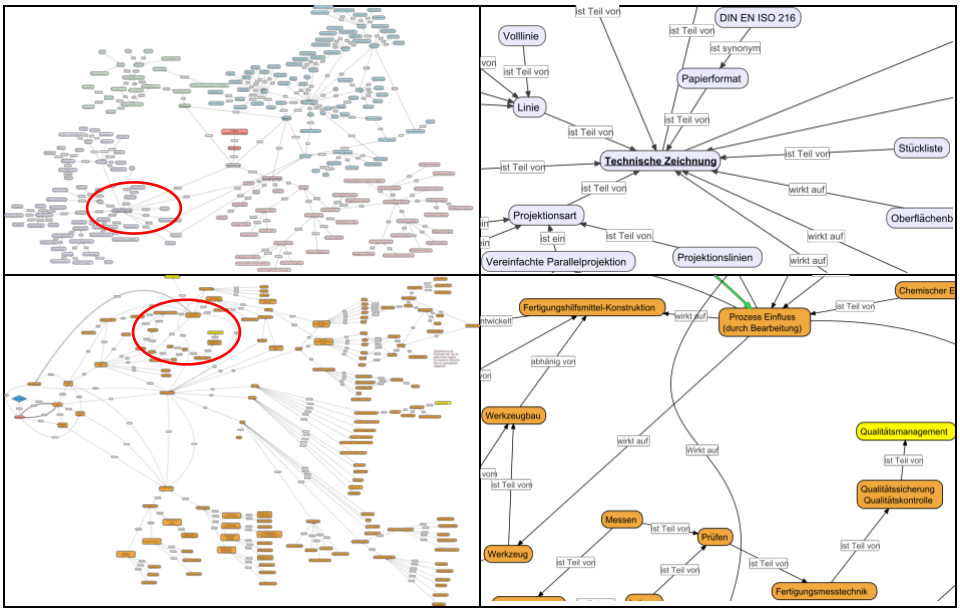

Fig. 4: Domain graphs (left) and detail (right) for engineering design (top) and production technology (bottom), from appendices of [Sc19] and [Bu19]

The participants reflected in their project reports:

A tendency was observed to "loose oneself in single branches of the network, explicating all too detailed and specific". So the focus should be kept high-level at an early stage. "Choice of terms should be self-explaining and comprehensive", with "standard technical terms wherever possible." [Bu19]

The main inhibiting factor for modelling was the lack of facility of navigation. As the semantic network grew, the planar graphic arrangement became more critical, intricate, and hindering to keep track and to maintain by zoom and pan functions. Extensive time, concentration, and reviewing had to be devoted to self-orientation and, after pauses, re-orientation in the data, resulting e.g. in duplicates [Sc19]. Automatic layout functions resulted in a counterintuitive arrangement and produced overlapping vertices and edges [Sc19]. Best overview resulted from manual layout that was cumbersome to handle because there was no reasonable coordination between the view of the model and the focus of exploration or editing. E.g. it was difficult to isolate selected areas for focused analysis and editing: Dragging vertices temporarily to the side (with the rest of the graph unmodified) resulted in promiscuous edge runs across the network. "Three-dimensional browsing and editing" would be considered a significant improvement. [Bu19] {This refers more to *dynamic* than actual *3D*.}

VUE allows structuring the graph into optional layers. Defining the layers "properties", "technologies", "processes", and "objects" turned out to be very helpful for populating, browsing, and editing the graph systematically [Sc19], without structuring the model strictly (like a taxonomy of these categories would). Some semantic tools offer a similar functionality of *perspectives* on the data to mask aspects of a graph that are irrelevant for the task at hand. In the *engineering design* graph, the coloring function was applied to sections for better orientation (Fig. 4, top left.)

The graphic presentation of a graph requires formal conventions or guidelines to transport meta-concepts about graph elements. E.g. in the production technology graph, the decision symbol from the Unified Modelling Language (UML) was used at first as a format to highlight link points to super- or other subdomains – This may confuse a reader with UML-background [Bu19]. For further work, edge types, building and structuring of the graph, and degree of detailing / branching out should be defined and aligned in the data [Sc19]. VUE's lack of a class model or type schema resulted in a difficult-to-handle manual alignment of edge types. [Sc19] {sic – During the study, it was not recognized that VUE actually can support external ontology class references e.g. for edge typing. However, this function requires an external type ontology and is not visualized descriptively - So after choosing an edge type, the edge would still have to be named consistently for display.}

The long-term target of making this data available and editable in the semantic web cannot be realized within VUE. [Sc19] {On later reflection, this should mainly be attributed to a lack of appropriate navigation.}

6 Implications for Strategies and Tools

The HyperCMS modelling approach yet has to be tested for full composition modelling with actual *disposition* hyperedges. Yet, exploration so far encourages the assumption that distributed competence data can be generated, connected, and queried using the HyperCMP modelling approach with hypergraphs. Modelling the domain first, and later supplementing disposition types was accidental but would allow systematic staging of the modelling process or division of work into lexical and compositional modelling as suggested in [Da20]. This also encourages confidence that competence modelling can build on previously developed knowledge graphs.

The key enabler for managing this modelling approach is a capable editor for non-computer domain experts to comfortably create, maintain, and utilize hypergraph data without coding. The software categories *ontology editors*, *graph databases*, *graph visualizations*, and *graph drawing tools* all allow semantic navigation and – except most visualization tools – editing. But they were designed for different use case (expert-supervised schema, API-driven data population, Big Data analysis and presentation, and diagramming of limited concepts). None currently seems to solve the essential functionality premised for HyperCMP modelling, not by itself, nor in combination with others. The closest candidate found was Grakn Core + Grakn Workbase, with AllegroGraph/Gruff or maybe Hyperscape as previously overlooked but promising candidates for further exploration.

Overall, a case can be made for a proposed software solution for the use case of decentrally managed, federated semantic modelling of competencies and similar concepts that:

- stores and publishes semantic graphs with native reification / hypergraph support,
- provides an integrated and intuitive graphic ("click-drag-and-label"-)interface for visualization, navigation, and editing of the data layer ,
- makes large data sets navigable by a combination of automatic layout following the focus of exploration and editing, preferably a force-directed layout, plus perspectives/filters to reduce complexity when building, maintaining, and exploring hypergraphs.
- allows decentrally managed, federated semantics for lexical modelling of domain concepts, compositional modelling of competencies, as well syntactical modelling of edge types (for lexical and for compositional modelling, as described in section 5) – ideally in one universal interface.
- does not bind types inherently to a taxonomy or an explicit schema. According to previously published guidelines, "taxonomies, categories, levels, and other assortative information must not be coded into the model {meta-model or pattern} itself, but as relationships into the semantic network data" [Da19].

7 Acknowledgements

This paper would not have been possible without the committed study participation of my students Philipp Burgdorf and Anna Schulenburg.

Bibliography

[Ca05] Carroll, J. J. et al.: Named graphs. In Journal of Web Semantics, 2005, 3; pp. 247–267. DOI: 10.1016/j.websem.2005.09.001

[Da06] Dahlmeyer, M. (at that time under the previous name Meyer): Management von Ingenieurkompetenzen im Spannungsfeld beruflicher Arbeitsteilung. Dissertation at the TU Berlin (University of Technology), 2006.

[DRS16] Dahlmeyer, M.; Reinhardt, K.; Schnauffer, H.-G.: Die Zukunft des Kompetenzmanagements. Vorschlag für ein transversales „System-Modell“ zur Kompetenzvernetzung zwischen Wissenschaft und Industrie als Voraussetzung für effektive Wissensarbeit. In: Gesellschaft für Wissensmanagement e.V.: gfwm Themen spezial. Frankfurt am Main, 2016.

[Du18] Dudáš, M. et al.: Ontology visualization methods and tools: a survey of the state of the art. In The Knowledge Engineering Review, 2018, 33.

[HT14] Hartig, O.; Thompson, B.: Foundations of an Alternative Approach to Reification in RDF, Technical Report, 2014. Available online: https://arxiv.org/pdf/1406.3399, accessed: 20/07/2020.

[BK19] Konert, J., Buchem, I. & Stoye, J.. Digitales Kompetenzverzeichnis mit Technologien des Semantic Web. In: Schulz, S. (Hrsg.), Proceedings of DELFI Workshops 2019. Bonn: Gesellschaft für Informatik e.V.z. (S. 61), 2019. DOI: 10.18420/delfi2019-ws-108.

[Bu19] Burgdorf, P.: Maschinenbau goes Web 2.0 – Inhalte im Semantic Web (Fertigungstechnik). Report (unpublished) from an interdisciplinary study project with Prof. Dr.-Ing. Matthias Dahlmeyer, Hochschule für Technik und Wirtschaft Berlin – University of Applied Sciences, 2019.

[Fi20] The emerging landscape for distributed knowledge, ontology, semantic web, knowledge base, graph based technologies and standards (June 2020). https://www.linkedin.com/pulse/emerging-landscape-distributed-knowledge-ontology-semantic-figay (lookup: July 2, 2020).

[Sc19] Schulenburg, A.: Maschinenbau goes Web 2.0 – Inhalte im Semantic Web (Konstruktion). Report (unpublished) from an interdisciplinary study project with Prof. Dr.-Ing. Matthias Dahlmeyer, Hochschule für Technik und Wirtschaft Berlin – University of Applied Sciences, 2019.