Valentina Ivanova, Tomi Kauppinen, Steffen Lohmann,
Suvodeep Mazumdar, Catia Pesquita, Kai Xu (Eds.)

VISUAL 2014

Proceedings of the International Workshop on
Visualizations and User Interfaces for Knowledge
Engineering and Linked Data Analytics

19th International Conference on Knowledge Engineering and
Knowledge Management (EKAW 2014)
November 24-28, 2014, Linköping, Sweden

Title: Proceedings of the International Workshop on Visualizations and User Interfaces for Knowledge Engineering and Linked Data Analytics (VISUAL 2014)

Editors: Valentina Ivanova, Tomi Kauppinen, Steffen Lohmann, Suvodeep Mazumdar, Catia Pesquita, Kai Xu

ISSN: 1613-0073

CEUR Workshop Proceedings
(CEUR-WS.org)

Preface

With data continuously generated as a result of daily activities within organizations and new data sources (sensor streams, linked datasets, etc.) introduced within knowledge management, the growth of information is unprecedented. Providing knowledge engineers and data analysts with visualizations and well-designed user interfaces can significantly support the understanding of the concepts, data instances, and relationships in different domains.

The development of appropriate visualizations and user interfaces is a challenging task, given the size and complexity of the information that needs to be displayed and the varied backgrounds of the users. Further challenges emerge from technological developments and diverse application contexts. There is no "one size fits all" solution but the various use cases demand different visualization and interaction techniques. Ultimately, providing better visualizations and user interfaces will foster user engagement and likely lead to higher-quality results in different areas of knowledge engineering and linked data analytics.

The workshop is divided into two half-day tracks, each focusing on one of the two workshop themes: The first track addresses visualizations and user interfaces as an integral part of knowledge engineering. They help to bridge the gap between domain experts and data management, and are essential to handle the increasing diversity of knowledge that is being modeled in ontologies, ensuring that it is easily accessible to a wide community. As knowledge-based systems and ontologies grow in size and complexity, the demand for comprehensive visualization and optimized interaction also rises.

A number of knowledge visualizations have become available in recent years, with some being already well-established, particularly in the field of ontology development. In other areas of knowledge engineering, such as ontology alignment and debugging, although several tools have recently been developed, few have a user interface, not to mention navigational aids or comprehensive visualization techniques. Other activities, such as data integration, rely on the relationships between the concepts of different ontologies, which not only multiplies the number of objects to be displayed but also compounds the problem with the portrayal of different kinds of relationships between concepts.

The second track addresses visualizations and user interfaces for linked data analytics. New and traditional knowledge practices, digitization of organizational processes, high performance computing and affordable datastores create an unprecedented amount of data as a part of daily organizational activities, at break-neck speed in a variety of formats. Conventional systems struggle to capture, store and analyze such dynamic and large scale data continuously generated. On its own, raw data has little value, but its value and significance is only unleashed when the data is extracted, processed and interpreted.

Visual Analytics attempts to address this challenge by harmoniously combining the strengths of human processing and electronic data processing. While semi-automated processes result in generating visualizations, humans can use visual processing and interactions to quickly identify trends, patterns and anomalies from large volumes of visual data. The growing challenges of analyzing big data, social media, linked data, and data streams have created an excellent opportunity for research in Visual Analytics.

The call for papers attracted high-quality submissions, and each paper was reviewed by at least three members of the program committee. Based on the reviews, we selected six papers for presentation at the workshop, which are included in this proceedings volume. We also reserved time for discussions and a demo session, to encourage dialogue and identify current and future challenges in the topic areas.

We thank all authors for their contributions and the members of the program committee for their valuable work in reviewing the submissions. We are also grateful to the workshop chairs Eva Blomqvist and Valentina Presutti as well as the general chairs Patrick Lambrix and Eero Hyvönen for their suggestions and support with the organization of this workshop.

November 2014

Valentina Ivanova
Tomi Kauppinen
Steffen Lohmann
Suvodeep Mazumdar
Catia Pesquita
Kai Xu

Program Committee

Marius Brade, TU Dresden

Amparo Cano, The Open University

Isabel Cruz, University of Illinois at Chicago

Aba-Sah Dadzie, University of Birmingham

Anna Lisa Gentile, University of Sheffield

Willem van Hage, SynerScope

Eero Hyvönen, Aalto University & University of Helsinki

Bum Chul Kwon, University of Konstanz

Patrick Lambrix, Linköping University

Enrico Motta, The Open University

Paul Mulholland, The Open University

Stefan Negru, MSD IT Global Innovation Center

Heiko Paulheim, University of Mannheim

Silvio Peroni, University of Bologna & CNR-ISTC

Chris Rooney, Middlesex University

Harald Sack, University of Potsdam

Gem Stapleton, University of Brighton

Margaret-Anne Storey, University of Victoria

Vojtěch Svátek, University of Economics, Prague

Cagatay Turkay, City University London

Stuart Wrigley, University of Sheffield

Leishi Zhang, Middlesex University

Contents

A Vision for Diagrammatic Ontology Engineering

Gem Stapleton[1], John Howse[1], Adrienne Bonnington[2], and Jim Burton[1]

[1] University of Brighton, UK,
{g.e.stapleton,john.howse,j.burton}@brighton.ac.uk,
[2] Horizons Regional Council, NZ,
adrienne.bonnington@horizons.govt.nz

Abstract. Ontology engineering is becoming more important, given the rapid rise of data in this information age. To better understand and reason about this data, ontologies provide us with insight into its structure. With this comes the involvement of a wide range of stakeholders, such as information analysts, software engineers, lawyers, and domain experts, alongside specialist ontology engineers. These specialists are likely to be adept at using existing approaches to ontology development, typically description logic or one of its various stylized forms. However, not all stakeholders can readily access such notations, which often have a very mathematical style. Many stakeholders, even including fluent ontology engineers, could benefit from visual approaches to ontology engineering, provided those approaches are accessible. This paper presents ongoing research into a diagrammatic approach to ontology engineering, outlining key research advances that are required.

Keywords: ontology engineering, diagrams, visualization

1 Introduction

Ontologies are used as structural frameworks for organizing information and are a form of knowledge representation. Ontology engineering - the process of producing ontologies - is becoming increasingly important and ubiquitous in society as a way of understanding the vast deluge of data that now exists in this information age. Notable examples include the semantic web, medicine (including SAPPHIRE for public health, SNOMED-CT for healthcare) and bioinformatics (e.g. the gene ontology GenNav). Indeed, there are various large repositories of freely available ontologies such as the Manchester OWL Corpus which contains over 4500 ontologies including, between them, over 3 million axioms [4], the Swoogle repository contains over 10000 ontologies [7] and BioPortal contains nearly 400 ontologies from the biomedical domain [3].

Ontology engineering requires the involvement of domain experts, who need to be able to communicate domain knowledge to ontology engineers. In turn, ontology engineers need to verify ontological choices with the domain experts and amend or refine the ontology based on feedback. Thus, the process of producing ontologies is very much bidirectional and depends on cross-disciplinary

communication. Barriers to communication can lead to errors, compounding an already difficult task.

Description logics and OWL are the commonly used (symbolic) notations for ontology engineering, and the latter is often seen as a stylized form of the former; the W3C's most recent specification of the web ontology language, called OWL 2.0 [5] is a decidable description logic [9]. Description logics promote a hierarchical taxonomy of concepts (called classes in OWL) as a primary modelling feature. Individuals can be members of concepts. Binary relations over concepts, called roles in description logic and properties in OWL, are used to relate individuals and concepts. In particular, individuals, concepts and roles form the vocabulary over which the axioms that form the ontology are defined. The term description logic actually refers to a family of logics, with the simplest such logic of significance being called $\mathcal{ALC}$ and perhaps the most practically used being called $\mathcal{SHOIN}^{(\mathcal{D})}$. The latter description logic is supported in the prominent ontology engineering tool called Protégé [6], which has a strong community of users including academics, governments and corporations [6].

As well as Protégé, other software tools have been implemented to support the creation of symbolically specified ontologies. However, when people meet to develop conceptual structures, including models of knowledge intended to become ontologies, they sometimes quickly move to sketching images to communicate their thoughts; see Howse et al. [16]. The inaccessibility of the DL family of symbolic notations is recognised by Rector et al. [18]: "Understanding the logical meaning of any description logic or similar formalism is difficult for most people, and OWL-DL is no exception."

Warren et al. back this insight up with an empirical study of the most commonly used OWL constructs, including class subsumption, disjointness and equivalence alongside All Values From and other role restrictions [24]. They found that "despite training, users are prone to certain misconceptions" and they observed that "one-third of [participants] commented on the value of drawing a diagram ... In the context of [description logics], diagrams offer a strategy to overcome misconceptions and generally support reasoning." Warren et al. further recognise that existing visualizations are "chiefly aimed at viewing the structure of the overall ontology or parts of the ontology rather than the more cognitively difficult features of Description Logics" and they identify that concept diagrams [16] are the exception. A major aim of our research is to address this identified shortcoming of symbolic notations by providing a diagrammatic alternative based on concept diagrams.

This paper presents an overview of the challenges that must be addressed in order to deliver a practical framework for using concept diagrams for ontology engineering. Key challenges include

1. devising a novel pattern-driven diagrammatic method for ontology engineering,
2. producing a heterogeneous logic that allows diagrams and symbols to be used in combination, and
3. developing techniques for automatically visualizing symbolically ontologies.

Throughout this paper we discuss the advances necessary to address these challenges. To begin, we present a brief overview of the state-of-the-art in ontology visualization research. We then discuss the design of concept diagrams, showing how they reflect known theories of what constitutes a good diagram. This motivates them as a suitable choice of notation for approaching the challenges set out above. We recognise the need for extensive empirical evaluations to ensure that concept diagrams are an accessible alternative to notations such as description logic, but for space reasons we do not discuss this further.

Our ambition is not only to visualize ontologies, but to enable them to be engineered and maintained visually, placing diagrams on an equal footing with traditional approaches. Throughout the paper, we illustrate key points using a real world ontology concerning cemeteries and burials. Ontology engineers interested in using concept diagrams are able to download a free practical user guide from http://www.ontologyengineering.org, under downloads.

2 State-of-the-Art Ontology Visualization Research

Ontology visualization improves access to information by ontology engineers, domain experts and other end-users. Benefits of visualization include revealing information that could be somewhat hidden, or unapparent, when using traditional symbolic notations. Such benefits have long been recognised in the related area of software engineering, as has the need to ensure that software models (akin to ontologies) are accessible to as wide a range of stakeholders as possible [8]. Katifori et al. survey the state-of-the-art in ontology visualization techniques up to 2007 [17], with examples including OWLViz [2], OntoGraf [1], and CMap Tools [14]. Like OWLViz, OntoGraf exploits node-link diagrams (also called graphs) to visualize subsumption relationships between concepts, but does not depict disjointness relationships. There is potential, therefore, for confusion to arise because of the partial information given. CMap Tools also exploit node-link diagrams and, as with OntoGraf, uses directed edges (arrows) to represent both subsumption relationships and role restrictions. Consequently, the saliency of these two different types of information is significantly reduced.

An example produced using CMAP Tools can be seen in figure 1. The fact that each Deceased is also a Person is asserted by the arrow labelled is a between Deceased and Person. The fact that each Deceased has a memorial of some sort is asserted by the use of the same syntactic device: the arrow labelled hasMemorial between Deceased and Memorial. CMaps are also capable of depicting disjointness information via arrows, though none is shown in figure 1. Thus, node-link-based visualizations such as CMaps and OntoGraf use arrows for role restrictions, subsumption and (where these can be depicted) disjointness relationships.

Similarity theory tells us that saliency is an important visual factor and, in particular, that different syntactic devices should represent different types of information [11]. This is because when visually searching for particular types of information, increasing degrees of similarity between the *target* syntax (rep-

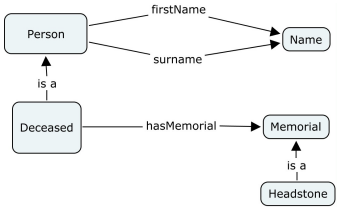


Fig. 1. A CMap visualization.

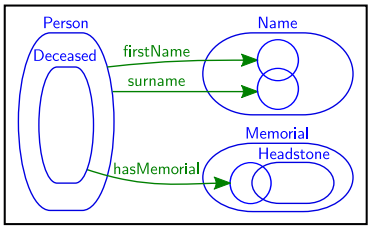


Fig. 2. A concept diagram.

resenting the required information) and *distracter* syntax (representing other information) leads to an increase in the time taken to perform tasks.

Concept diagrams aid information saliency by avoiding the use of identical syntactic types for different informational types: dots (or, in general, node-link diagrams), closed curves, and arrows represent individuals, concepts and roles respectively. An example concept diagram, expressing similar information to the CMap visualization, is in figure 2. The placement of (the curve labelled) Deceased inside Person expresses that Deceased is subsumed by Person. Since Person and Memorial do not overlap, the diagram also expresses that these two concepts are disjoint (have no individuals in common). The arrow labelled firstName, sourced on Person and targeting the unlabelled curve inside Name asserts that people's first names are all names (in DL, this is an All Values From restriction: $\mathsf{Person} \sqsubseteq \forall \mathsf{firstName}.\mathsf{Name}$). Here, the unlabelled curve represents an anonymous concept containing precisely the names that are people's first names. The other two arrows are interpreted in the same way.

One visualization technique which was developed specifically as a diagrammatic logic equivalent to the simple description logic $\mathcal{ALC}$ is a variation on existential graphs by Dau and Eklund [10]. An example of an existential graph is given in figure 3 that expresses the All Values From restriction $\mathsf{Person} \sqsubseteq \forall \mathsf{firstName}.\mathsf{Name}$. The existential graph literally translates to 'it is not the case that there is a person who has a first name that is not a Name'; curves represent negation, lines represent anonymous individuals and the written words correspond to concepts and roles. Existential graphs are not readily accessible since their syntax is somewhat restrictive: they have the flavour of a minimal first-order logic with only existential quantification, negation and conjunction.

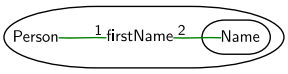


Fig. 3. An existential graph.

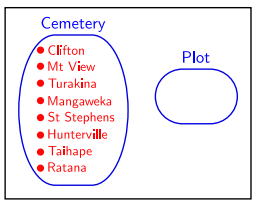


Fig. 4. Representing individuals.

3 The Design of Concept Diagrams

The design of concept diagrams has been informed by theories as to what constitutes an effective visualization. We have already seen that similarity theory leads us to use different syntactic items for different kinds of constructions. This is consistent with the Gestalt principle of *good form*, which tells us that there is a tendency for people to group together graphical objects that share some property, such as colour or shape. In concept diagrams, individuals are visualized using nodes, concepts by closed curves and role restrictions by arrows. We have already seen the use of curves and arrows; figure 4 shows how individuals are represented. This diagram asserts that Clifton is a Cemetery but not a Plot, with the same being true of the other seven individuals. Using different types of graphical objects ensures the same type of syntax is used to represent the same type of semantic property. Further, we use shape to partition the curves into two distinct sets: those which represent named concepts and, respectively, unnamed concepts are represented using rounded rectangles and, respectively, circles.

Similarly, graphical objects that have different properties will typically be considered as being in different groups by people. This suggests that different semantic constructs should be represented by different graphical objects, consistent with similarity theory. Colour could also be used to good effect, if we want to give visual clues about some semantic commonality or distinctness. We have adopted the convention, when colouring concept diagrams, that nodes, curves, and arrows are assigned different colours (in this paper, red, blue and green respectively). Rectangles, which identify diagram boundaries, are all black.

The way in which concept diagrams use this syntax to make statements has been carefully considered. In particular, the spatial relationship between curves reflects the semantic relationships that they convey: concept subsumption, disjointness and intersection are represented by enclosure, disjointness and overlap of curves respectively. Likewise, the location of nodes either inside or outside of curves corresponds to the individuals represented being either members of, or not members of respectively, the corresponding concepts. Properties are directional relationships which are captured by arrows, indicating direction. These types of features, embodied in concept diagrams' design, are known as *well-matched* [13].

An important feature of concept diagrams is that they are fully formalized [22]. As is standard with visual languages [12, 15], concept diagrams are formalized using an abstract syntax (sometimes called type syntax). Their semantics are defined using a standard model-theoretic approach, akin to that used for description logics. Due to space reasons, we refer the reader to [22] for the full details on the formalization of concept diagrams.

4 Patterns

We believe that concept diagrams are able to *readily* express commonly occurring symbolic axioms, for which we have devised a set of simple diagrammatic patterns [23]. Such an approach is thought to aid building ontologies (i.e. defining

the axioms), since the patterns will, essentially, be standard templates for commonly occurring constraints. However, using a separate diagram for each piece of information does not allow the exploitation of many important diagrammatic features. In particular, single diagrams that express rich information can do so in an accessible way and such diagrams can correspond to many symbolic axioms (discussed further in section 5). To this end, we demonstrate how simple diagrammatic patterns can be merged to produce informationally rich diagrams.

4.1 Simple Patterns

A series of simple patterns was presented in [23], expressing concept subsumption and disjointness as well as All Values From and Some Values From role restrictions. Here, we illustrate eight of these simple patterns.

Figures 5 and 6 show instances of patterns for asserting that individuals are and, respectively, are not members of concepts. In particular, figure 5 asserts that M (short for 'male') is a Gender whereas figure 6 asserts that M is not an Occupation. Figures 7 and 8 are instances of the concept subsumption and concept disjointness patterns. The former figure tells us that Deceased is subsumed by Person whereas the latter expresses that Person and Gender are disjoint.

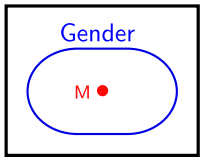


Fig. 5. Inclusion.

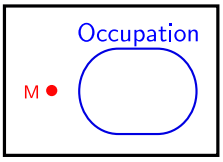


Fig. 6. Exclusion.

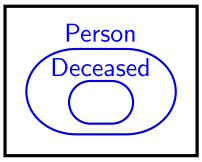


Fig. 7. Subsumption.

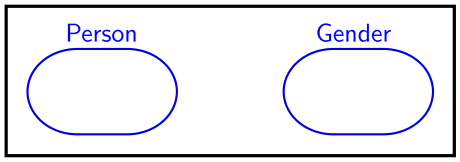


Fig. 8. Disjointness.

Turning our attention to roles, figure 9 diagrammatically expresses an All Values From restriction. In particular, it tells us that under the role gender, Person has all values from the concept Gender. This concept diagram illustrates the use of multiple boundary rectangles. The spatial relationships between syntactic items are only of semantic importance within innermost boundary rectangles. In this case, the two innermost rectangles mean that the diagram does *not* tell us that Person and Gender are disjoint (even though this happens to be true, given the information in figure 8).

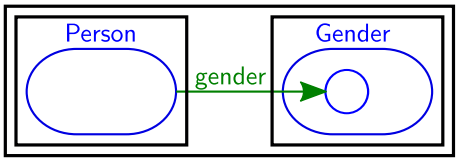


Fig. 9. All Values From.

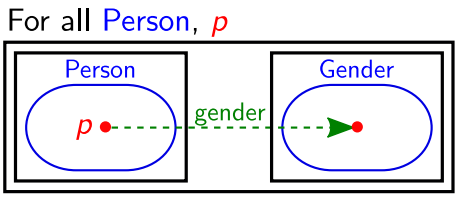


Fig. 10. Some Values From.

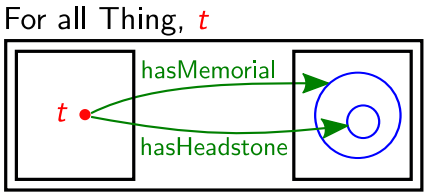


Fig. 11. Role subsumption.

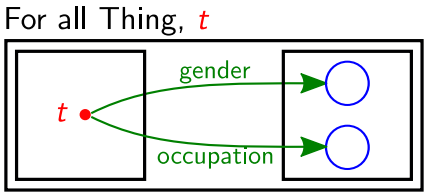


Fig. 12. Role disjointness.

The Some Values From pattern used in figure 10 (which is different from the Some Values From pattern in [23]) illustrates two further features of concept diagrams: dashed arrows and the use of quantifiers to talk about anonymous individuals. This concept diagram makes an assertion about all individuals that are people, using the *quantification expression* For all Person, p. The *dashed* arrow targets an anonymous individual that is a Gender. This dashed arrow tells us that the individual, p, is related to *at least* the individual at the target. That is, the person p has at least one gender.

Patterns for role subsumption and role disjointness are instantiated in figures 11 and 12 respectively. As with concept subsumption and disjointness, they also exploit the topological properties of containment and disjointness to express the required information. For example, figure 11 tells us that, for each thing t, the set of t's headstones in subsumed by the set of t's memorials.

4.2 Merging Patterns

Using one diagram to express a small amount of information may be helpful when developing ontologies, but sometimes having diagrams containing rich information can give a better overview of how concepts and roles interact. We plan to develop methods of merging simple patterns together in order to produce richer diagrams. Such a merging process is illustrated in figures 13 to 15. Firstly, figure 8 tells us that Person and Gender are disjoint, meaning that we can delete the two innermost rectangles from figure 9. The result is in figure 13, which expresses the same information as the two original patterns. The other diagrams in figures 14 and 15 are similarly obtained; here the graph labelled M indicates that M is a Gender where the edge represents disjunction.

Using merging techniques, semantically rich diagrams that display significant proportions of ontologies can be produced. An example is given in figure 16,

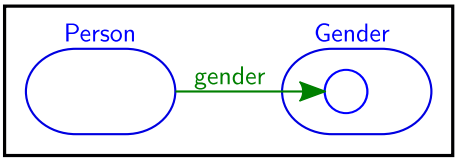


Fig. 13. Merging figures 8 and 9.

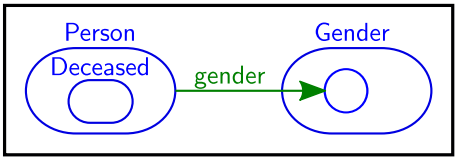


Fig. 14. Merging figures 13 and 7

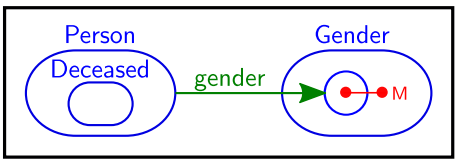


Fig. 15. Merging figures 14 and 5

which depicts a large fragment of a cemetery ontology. This diagram depicts 13 *inclusion* axioms, four *subsumption* axioms, 709 (binary) *disjointness* axioms and 46 *all values from* axioms. It is an interesting avenue of future work to determine which patterns can be merged, and how. For example, in figure 16 only axioms whose patterns do not include explicit quantification have been merged. A key aspect of this work will be to devise heuristics that identify diagrams that can be merged to produce effective visualizations. The fact that figure 16 depicts 772 axioms in a single readable diagram indicates that the notation scales to some reasonable extent; of course, screen real estate presents a scalability problem for both visual and symbolic notations.

5 Heterogeneous Ontology Engineering

Unlike many stakeholders, expert ontology engineers are fluent in the use of symbolic logics. The framework for diagrammatic ontology engineering that we envisage will allow ontologies to be engineered symbolically and diagrammatically and, as a by-product, allow the visualization of already developed symbolic ontologies. A major challenge is to allow diagrams and symbols to be used in tandem by providing a seamless integration of the two paradigms. This requires the two 'views' (diagrammatic and symbolic) of the ontology to be kept in sync, so any update made to the symbolic axioms is reflected diagrammatically and

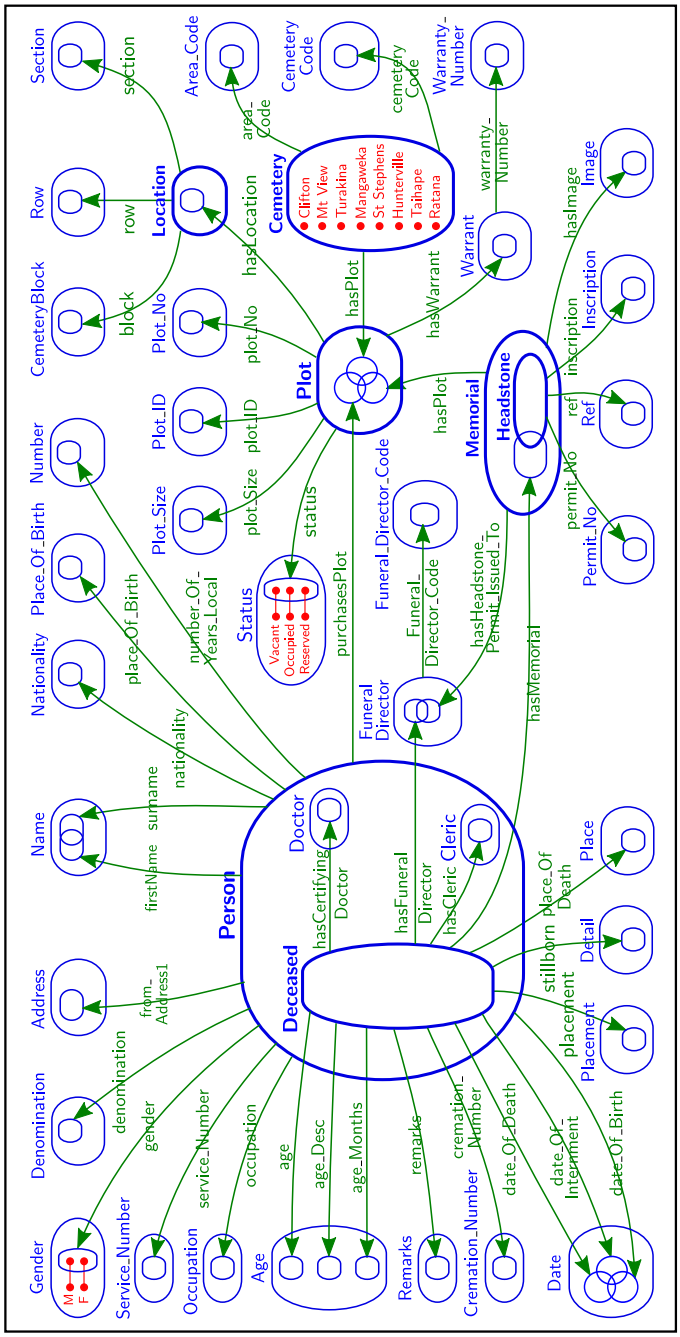


Fig. 16. A large fragment of the cemetery ontology.

vice versa; this idea of two views is illustrated using figure 15 and the DL axioms: Person $\sqsubseteq \forall$gender.Gender, Person $\sqcap$ Gender $\sqsubseteq \bot$, Deceased $\sqsubseteq$ Person, and Gender(M).

Core to solving this challenge is the derivation of effective translations between diagrammatic axioms and symbolic axioms. It is expected that the patterns described above, with the merging results, will form a basis for translating symbolically-specified ontologies into concept diagrams. However, significant advances are required, using patterns as a starting point. Requirements of the translation methods are likely to include, but not be limited to: visualize the entire ontology; visualize all concepts; visualize all axioms involving a particular concept or set of concepts; and visualize all axioms involving a particular role or set of roles.

Further, we expect to devise translations that permit multiple levels of visualization: top-level overviews, some means of exploring the overview including zooming and filtering, and drilling down to low-level detail as required. Lastly, when symbolic axioms are altered, as will often happen during the refinement and debugging phases of ontology development, the corresponding diagrammatic axioms must be updated. Translations from concept diagrams to description logic will also be devised but, again, finding an optimal symbolic set of axioms will be difficult; as a trivial example, one must choose between the axioms $C_1 \sqsubseteq C_2$ and $C_1 \sqsubseteq C_3$ and the single axiom $C_1 \sqcup C_2 \sqsubseteq C_3$.

6 Automated Layout

When devising heterogeneous approaches to ontology engineering, it is important that one is able to automatically draw – or lay out – concept diagrams. When automatically drawing diagrams, one starts with the abstract syntax of the required diagram or diagrams. The problem is to draw effective diagrams, with the given abstract syntax. There is considerable choice of layout, where figure 17 shows what we consider to be a bad layout of the diagram shown in figure 2.

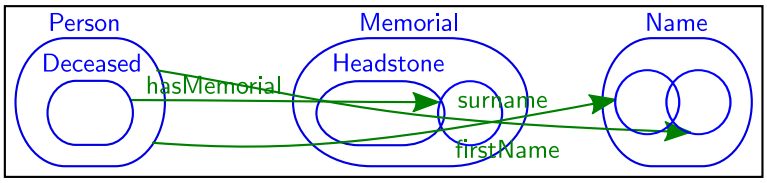


Fig. 17. A bad layout.

Indeed, the problem of drawing multiple diagrams, where common parts of two or more diagrams should – for effective use – look identical substantially increases the difficulty. This is illustrated in figures 18 and 19. Figure 18 presents, in

separate diagrams, some relationships involving the classes Memorial and Headstone. The two diagrams have been drawn so that they have a layout that reflects the structurally similar information represented. However, we expect that the diagrams in figure 19 are a more effective pair of visualizations than those figure 18, since their common parts have the same relative positioning in the plane.

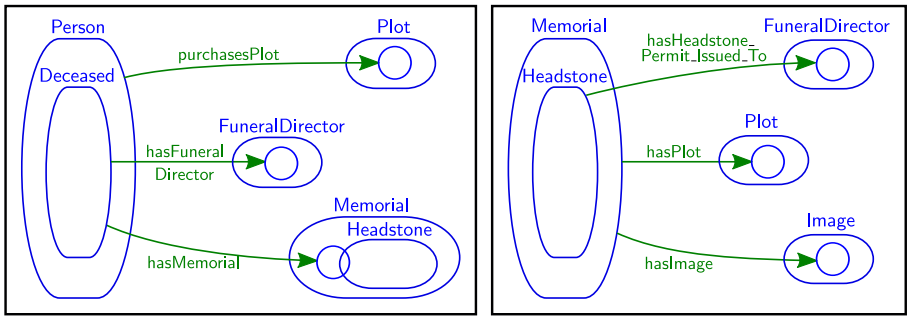


Fig. 18. Some Deceased relationships and some Memorial relationships.

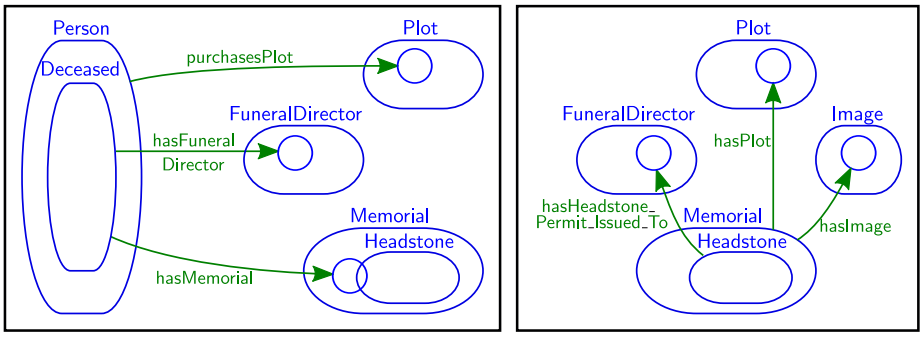


Fig. 19. Memorial relationships redrawn to align with layout of Deceased relationships

The problem of drawing multiple-diagrams when translating between notations, where many diagrams can arise from symbolic axioms. Whilst the drawing problem is inherently computationally complex, it is necessary to seek drawing and layout algorithms that have acceptable run times for most cases that arise in practice and that produce usable results. Empirical evaluations will be necessary to determine what constitutes an effective layout, informing choices about diagram topology and geometry.

As a staring point for a general drawing algorithm for concept diagrams, existing Euler diagram drawing methods such as [19–21] can be extended to simply add the extra syntax required (giving primacy to the Euler diagram). However,

this approach will require extensive modifications to existing algorithms, since the initially drawn Euler diagram may compromise the layout of the subsequently drawn syntax. Ideally, the drawing algorithms will consider the entirety of the to-be-drawn diagram when making layout choices, not assigning primacy to any one syntactic part. Moreover, general drawing algorithms are computationally expensive and may not always produce effective layouts. For this reason, we plan to devise, first, a layout algorithm for classes of concept diagrams that commonly arise in practice.

Lastly, drawing algorithms are required that incrementally update diagrams when edits are made to a symbolically specified ontology, to keep the two views in sync. These edits can include: deleting axioms, adding new axioms or amending existing axioms (which can be considered as a deletion followed by an addition). When an axiom is deleted, this can have profound effects on the layout of diagrams. For instance, deleting a disjointness axiom may require some of the (non-overlapping) curves in a diagram to be re-routed (so they overlap, thus not asserting disjointness). Similarly, adding an axiom can also require significant changes to parts of the diagrams, such as removing an overlap between curves. When diagrams include a lot of syntax, making such changes to curves is not necessarily straightforward.

7 Conclusion

Our vision is to produce a fully supported diagrammatic logic for ontology engineering, which includes implementing software to support its use. If successful, this diagrammatic logic will provide more accessible ways to develop, update and maintain ontologies. Since the diagrammatic framework that we envisage will be fully integrated with existing symbolic approaches, expert ontology engineers will be able to effectively collaborate with stakeholders who prefer a diagrammatic approach. We view this integration as necessary for the take-up of concept diagrams generally.

Acknowledgements We are indebted to Rangitikei District Council for providing us with access to the cemetery data on which the examples in this paper are based.

References

1. OntoGraf. http://protegewiki.stanford.edu/wiki/OntoGraf (2014), accessed Aug 2014
2. OWLViz. http://protegewiki.stanford.edu/wiki/OWLViz (2014), accessed Aug 2014
3. The BioPortal ontology repository. http://bioportal.bioontology.org/ (2014), accessed Aug 2014
4. The Manchester OWL Corpus. http://owl.cs.manchester.ac.uk/publications/ (2014), accessed Aug 2014

5. The OWL 2 Web Ontology Language. http://www.w3.org/TR/owl2-overview/ (2014), accessed Aug 2014
6. The Protégé web site. http://protege.stanford.edu/ (2014), accessed Aug 2014
7. The Swoogle ontology repository. http://swoogle.umbc.edu/ (2014), accessed Aug 2014
8. Atkinson, C., Kühne, T.: Model-Driven Development: A Metamodeling Foundation. IEEE Softw. 20, 36–41 (Sep 2003)
9. Baader, F., Calvanese, D., McGuinness, D., Nadi, D., (eds), P.P.S.: The Description Logic Handbook. CUP (2003)
10. Dau, F., Eklund, P.: A diagrammatic reasoning system for the description logic $\mathcal{ACL}$. Journal of Visual Languages and Computing 19(5), 539–573 (2008)
11. Duncan, J., Humphreys, G.W.: Visual search and stimulus similarity. Psychological review 96(3), 433 (1989)
12. Erwig, M.: Abstract syntax and semantics of visual languages. Journal of Visual Languages and Computing 9, 461–483 (1998)
13. Gurr, C.: Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. Journal of Visual Languages and Computing 10(4), 317–342 (1999)
14. Hayes, P., Eskridge, T.C., Mehrotra, M., Bobrovnikoff, D., Reichherzer, T., Saavedra, R.: Coe: Tools for collaborative ontology development and reuse. In: Knowledge Capture Conference (K-CAP). vol. 2005 (2005)
15. Howse, J., Molina, F., Shin, S.J., Taylor, J.: Type-syntax and token-syntax in diagrammatic systems. In: Proceedings FOIS-2001. pp. 174–185. ACM Press (2001)
16. Howse, J., Stapleton, G., Taylor, K., Chapman, P.: Visualizing ontologies: A case study. In: International Semantic Web Conference. pp. 257–272. Springer (2011)
17. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods a survey. ACM Comput. Surv. 39(4), 10+ (Nov 2007)
18. Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL Pizzas: Practical Experience of Teaching OWL-DL. In: Motta, E., Shadbolt, N., Stutt, A., Gibbins, N. (eds.) Engineering Knowledge in the Age of the Semantic Web, LNCS, vol. 3257, chap. 5, pp. 63–81. Springer (2004)
19. Riche, N., Dwyer, T.: Untangling Euler diagrams. IEEE Transactions on Visualization and Computer Graphics 16(6), 1090–1099 (2010)
20. Simonetto, P.: Visualisation of Overlapping Sets and Clusters with Euler Diagrams. Ph.D. thesis, Université Bordeaux (2012)
21. Stapleton, G., Flower, J., Rodgers, P., Howse, J.: Automatically drawing Euler diagrams with circles. Journal of Visual Languages and Computing 23(3), 163–193 (2012)
22. Stapleton, G., Howse, J., Chapman, P., Delaney, A., Burton, J., Oliver, I.: Formalizing Concept Diagrams. In: Visual Languages and Computing. pp. 182–187. Knowledge Systems Institute (2013)
23. Stapleton, G., Howse, J., Taylor, K., Delaney, A., Burton, J., Chapman, P.: Towards Diagrammatic Ontology Patterns. In: 4th Workshop on Ontology and Semantic Web Patterns. CEUR, Sydney, Australia (Oct 2013)
24. Warren, P., Mulholland, P., Collins, T., Motta, E.: The Usability of Description Logics. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) The Semantic Web: Trends and Challenges, LNCS, vol. 8465, pp. 550–564. Springer (2014)

OntoViBe: An Ontology Visualization Benchmark

Florian Haag[1], Steffen Lohmann[1], Stefan Negru[2], and Thomas Ertl[1]

[1]Institute for Visualization and Interactive Systems, University of Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany
{Florian.Haag,Steffen.Lohmann,Thomas.Ertl}@vis.uni-stuttgart.de
[2]Faculty of Computer Science, Alexandru Ioan Cuza University,
Strada General Henri Mathias Berthelot 16, 700483 Iasi, Romania
stefan.negru@info.uaic.ro

Abstract. A variety of ontology visualizations have been presented in the last couple of years. The features of these visualizations often need to be tested during their development or for evaluation purposes. However, in particular for the testing of special concepts and combinations thereof, it can be difficult to find suitable ontologies. We have developed OntoViBe, an ontology covering a wide variety of OWL 2 language constructs for the purpose of testing ontology visualizations. We describe the design principles underlying OntoViBe and present the supported features in coverage matrices. Finally, we load OntoViBe with ontology visualization tools and point to some noteworthy aspects of the respective visualizations that become apparent and demonstrate how OntoViBe can be used for testing ontology visualizations.

Keywords: Ontology, visualization, benchmark, evaluation, OWL.

1 Introduction

Developing and working with ontologies can be supported by ontology visualizations. Over the past years, a number of visualization approaches geared towards the peculiarities of ontologies have been proposed. Most of the available approaches use node-link diagrams to depict the graph structure of ontologies, while some apply other diagram types like treemaps or nested circles [7,9,11].

During the development of such ontology visualizations, testing with a variety of existing ontologies is required to ensure that the concepts from the underlying ontology language are adequately represented. The same needs to be done to determine the features of an ontology visualization and get an impression of how different ontology language constructs are visually represented. Still, repeatedly loading a set of ontologies that cover a wide variety of language constructs can be a tedious task, even more so as the most common constructs tend to appear over and over in each of the tested ontologies. In order to help that process with respect to ontologies based on the OWL 2 Web Ontology Language, we developed OntoViBe, an Ontology Visualization Benchmark.

Basically, OntoViBe is an ontology that has been designed to incorporate a comprehensive set of OWL 2 language constructs and systematic combinations thereof. While it is oriented towards OWL 2, it also includes the concepts of OWL 1 due to the complete backwards compatibility of the two ontology languages, i.e., all OWL 1 ontologies are valid OWL 2 ontologies [17].

As opposed to most other benchmarks found in the computing world, OntoViBe is not meant for testing the scalability of visualizations with respect to the number of elements contained in ontologies, but rather aims for the scope of visualizations in terms of supported features. Related to this, it focuses on the representation of what is usually called the TBox of ontologies (i.e., the classes, properties, and datatypes), while it does not support the testing of ABox information (i.e., individuals and data values), which is the focus of most related work.

2 Related Work

Several benchmarks for ontology tools have been developed in the past. One well-known benchmark in this area is the Lehigh University Benchmark (LUBM), published by the SWAT research group of Lehigh University [8]. It consist of three components: 1) an ontology of moderate size and complexity describing concepts and relationships from the university domain, 2) a generator for random and repeatable instance data that can be scaled to an arbitrary size, and 3) a set of test queries for the instance data as well as performance metrics.

Since the LUBM benchmark is bound to the university domain, the SWAT research group developed another benchmark that can be tailored to different domains [18]. It uses a probabilistic model to generate an arbitrary number of instances based on representative data from the domain in focus. As an example, the Lehigh BibTeX Benchmark (LBBM) has been created with the probabilistic model and a BibTex ontology. Another extension of LUBM has been proposed with the University Ontology Benchmark (UOBM) [12]. UOBM aims to include the complete set of OWL 1 language constructs and defines two ontologies, one being compliant with OWL Lite and the other with OWL DL. Furthermore, it adds several links to the generated instance data and provides related test cases for reasoners.

All these benchmarks focus primarily on performance, efficiency, and scalability, but do not address the visual representation of ontologies. Furthermore, they are mainly oriented towards instance data (the ABox), while systematic combinations of classes, properties, and datatypes (the TBox) are not further considered. Even though UOBM provides comparatively complete TBox information, it has been designed to test OWL reasoners and not ontology visualizations. This is also the case for JustBench [4], which uses small and clearly defined ontology subsets to evaluate the behavior of OWL reasoners.

There are also some benchmarks addressing specific aspects of ontology engineering. A number of datasets and test cases emerged, for instance, as part of the Ontology Alignment Evaluation Initiative (OAEI) [1]. A related dataset

has been created in the OntoFarm project, which provides a collection of ontologies for the task of of testing and comparing different ontology alignment methods [16]. An extension of the OntoFarm idea is the MultiFarm project, which offers ontologies translated into different languages with corresponding alignments between them [13]. Overall, the test cases are intended to evaluate and compare the quality and performance of matching algorithms, in the latter case with a special focus on multilingualism.

The W3C Web Ontology Working Group has also developed test cases for OWL 1 [6] and OWL 2 [15]. They are meant to provide examples for the normative definition of OWL and can, for instance, be used to perform conformance checks. However, there is not yet any benchmark particularly addressing the visualization of ontologies to the best of our knowledge. To close this gap, we developed OntoViBe, which will be described in the following.

3 Ontology Visualization Benchmark (OntoViBe)

The structure and content of OntoViBe is based on the OWL 2 specifications [17], with the following requirements:

- A wide variety of OWL 2 language constructs must appear. This includes constructs such as class definitions or different kinds of properties, as well as modifiers for these, such as *deprecated* markers.
- Subgraphs that represent compound concepts must appear. This includes small groups of classes that are coupled by a particular property.

Moreover, we tried to keep the overall ontology as small as possible in number of elements. Like this, rather than a mere enumeration of the elements and concepts supported by OWL 2, chances are that the ontology can be completely displayed and grasped "at a single glance" and thereby convey a complete impression of the features supported by the visualization being examined.

OntoViBe was assembled by creating an instance of each of the subgraph structures. Where possible, classes were reused to keep the ontology small. For instance, to include the OWL element *object property*, a subgraph structure consisting of two classes connected by an object property was added. Hence, two classes were inserted into the ontology, and an object property that uses either of the two classes as its domain and range, respectively, was defined. Furthermore, the element *datatype property* needed to appear in the ontology. A compact subgraph structure to express that element consists of a class linked to a datatype property. As the class does not need to have any specific characteristics of its own, one of the two previously inserted classes could be reused.

After that, some of the existing elements were modified to cover all elements and features that we wanted to consider at least once in the ontology. For example, some properties were declared as *functional* or *deprecated*. For any element type that still did not appear in the ontology, a minimal number of extra classes were added (the addition of extra properties and datatypes was not necessary).

Listing 1.1. Concepts based upon set operators are featured in two variants, a small set with two elements, and a larger one with more elements.

```
30 this:UnionClass a owl:Class ;
31   owl:unionOf ( this:Class1 this:DeprecatedClass ).
32
33 this:LargeUnionClass a owl:Class ;
34   owl:unionOf ( this:UnionClass other:ImportedClass this:PropertyOwner ).
```

Listing 1.2. OntoViBe defines custom OWL data ranges.

```
94 this:DivisibleByFiveEnumeration a rdfs:Datatype ;
95   owl:equivalentClass [
96     a rdfs:Datatype ;
97     owl:oneOf ( 5 10 15 20 )
98   ].
99
100 this:UnionDatatype a rdfs:Datatype ;
101   owl:unionOf ( this:DivisibleByTwoEnumeration this:
        DivisibleByFiveEnumeration ).
```

Lastly, all elements in the ontology were named in a self-descriptive manner to allow for an easier interpretation and analysis. For instance, a deprecated class is called `DeprecatedClass`, while the larger of the union classes is called `LargeUnionClass`.

3.1 Exemplary Parts of OntoViBe

Many of the structures could be added in a straightforward way. In some cases, further considerations were required to adequately address the more flexible features of OWL.

Concepts defined based upon set operators (*unionOf*, *intersectionOf*, *complementOf*) come in two variants. One of them uses a set comprising two elements as an example for a small set, while the other features more set elements, usually three (Listing 1.1).

OntoViBe also includes OWL data ranges (Listing 1.2). Visualizations may or may not represent the exact definitions of these data ranges, but even if they do not, support for datatype properties with custom data ranges needs to be tested. Therefore, custom data ranges are used by some datatype properties in OntoViBe, while common datatypes are used for most other properties (Listing 1.3).

In order to check how imported ontology elements are treated, OntoViBe consists of two components. The core ontology[1] contains most of the definitions, but a few classes, properties, and datatypes are defined in an additional module[2], whose content is imported into the core ontology (Listing 1.4).

[1] `http://ontovibe.visualdataweb.org/1.0#`

[2] `http://ontovibe.visualdataweb.org/1.0/imported#`

Listing 1.3. Both custom and common datatypes are used by properties.

```
this:standardTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range xsd:integer .

this:customTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range this:DivisibleByFiveEnumeration .
```

Listing 1.4. Some of the definitions are imported from a separate ontology module.

```
<http://ontovibe.visualdataweb.org/1.0#> a owl:Ontology ;
  owl:versionIRI <http://ontovibe.visualdataweb.org/1.0#> ;
  owl:imports <http://ontovibe.visualdataweb.org/1.0/imported#> ;
  <http://purl.org/dc/elements/1.1/title> "Ontology Visualization Benchmark
       (OntoViBe)" .
```

Listing 1.5. Sets of properties connected to the same classes allow for testing whether a visualization positions such properties in a non-overlapping way. This example shows two cyclic properties connected to the same class.

```
this:cyclicProperty2 a owl:ReflexiveProperty ;
  rdfs:domain this:MultiPropertyOwner ;
  rdfs:range this:MultiPropertyOwner .

this:cyclicProperty3 a owl:ObjectProperty ;
  rdfs:domain this:MultiPropertyOwner ;
  rdfs:range this:MultiPropertyOwner .
```

In ontology visualizations, sets of properties between the same pair of classes (or the same class and literal) pose a particular challenge, as they may lead to overlapping and thus illegible representations. Several of these cases have been considered in OntoViBe. For instance, Listing 1.5 shows two cyclic properties (i.e., properties whose domain and range are identical) connected to the same class.

Finally, a few of the ontology elements are provided with labels, to check how visualizations cope with multilingual labels that may also contain non-ASCII characters (Listing 1.6). For all non-ASCII characters, the escaped ASCII representation is used in the ontology file, as that maximizes the chances for a good compatibility with the parser reading the file.

3.2 Verification of Coverage and Omissions

To verify that OntoViBe covers most of the features defined by the OWL 2 specifications, we provide two coverage matrices. Table 1 juxtaposes the elements of OntoViBe with systematically listed OWL 2 features as described in the specifications. Table 2 shows which OntoViBe elements use which concrete OWL 2 identifiers, as per the IRIs declared in the OWL 2 Namespace Document [3].

Listing 1.6. Multilingual labels, some of which contain characters from different scripts, exist for a few of the ontology elements.

```
this:importedTypeDatatypeProperty a owl:DatatypeProperty ;
  rdfs:domain this:PropertyOwner ;
  rdfs:range other:DivisibleByThreeEnumeration ;
  rdfs:label "imported type datatype property"@en ;
  rdfs:label "propri\u00E9t\u00E9 d'un type de donn\u00E9es import\u00E9"
      @fr ;
  rdfs:label "\u4E00\u79CD\u5BFC\u5165\u7C7B\u578B\u7684\u6570\u636E\u7C7B\
      u578B\u6027\u8D28"@zh-Hans .
```

The tables also reveal some parts of OWL that are intentionally not included in OntoViBe:

Cardinalities: OntoViBe defines only a few cases of cardinality constraints for properties: Structurally, these can be distinguished as *no cardinality*, *cardinality on one end of a property relation* and *cardinality on both ends of a property relation*. Regarding the concrete cardinality constraints applied, exact cardinality, a minimum cardinality, and a combination of a minimum and a maximum cardinality are included in OntoViBe. Moreover, one of the cardinality constraints is qualified and thus applies only to instances of a specific class. These cases can thus only be used for checking whether cardinalities are displayed at all.
We have opted against integrating all supported cardinalities in OntoViBe, as the number of possible combinations would be considerable—in particular, when considering that "special values" such as zero and one might be displayed in special ways. The total number of properties in OntoViBe would have to be increased while providing only minor additional insight into the tested visualization.

Annotations: Informative metadata has no effect on the conceptual structure of an ontology, which is focused in OntoViBe. For that reason, only the most prevalent metadata attributes, such as labels or the ontology title, have been integrated into OntoViBe.

Equivalent constructs: In cases of conceptually equivalent ways to express statements in OWL, only one way was integrated into OntoViBe. For instance, deprecation of ontology elements can either be expressed by adding the `owl:deprecated` attribute or by declaring the element as belonging to one of the classes `owl:DeprecatedClass` or `owl:DeprecatedProperty`.

Deprecated elements: Deprecated language constructs of OWL itself are not used in OntoViBe. An example is `owl:DataRange` that has been deprecated as of OWL 2 in favor of `rdfs:Datatype` [3].

Moreover, statements referring to particular individuals have not been included, as OntoViBe focuses on visualizations of the TBox of ontologies.

4 Examples of Application

In the following, we demonstrate the usefulness of OntoViBe by applying four ontology visualizations to it: SOVA, VOWL, OWLViz, and OntoGraf. The latter two come with the default installation of the popular ontology editor Protégé [2] (desktop version 5.0), while the first two are comparatively well-specified with regard to the visual elements they are based on. Moreover, we analyze the ontology documentation generated for OntoViBe by the Live OWL Documentation Environment (LODE).

We present screenshots of all these ontology visualizations that give an impression of the supported features. We point out peculiarities of the visualization approaches and their implementations that become apparent based on OntoViBe. By this, we would like to provide some examples of how to use OntoViBe for the qualitative analysis of ontology representations, and to confirm that such an analysis is feasible by using OntoViBe.

It should be noted that a comprehensive analysis of ontology visualizations requires additional methods, such as a checklist comprising further evaluation criteria. These methods are typically not generic but tailored to the type of visualization. For instance, measures for graph visualizations of ontologies could include the total number of edges and edge crossings. However, such additional measures are outside the scope of this work.

4.1 SOVA

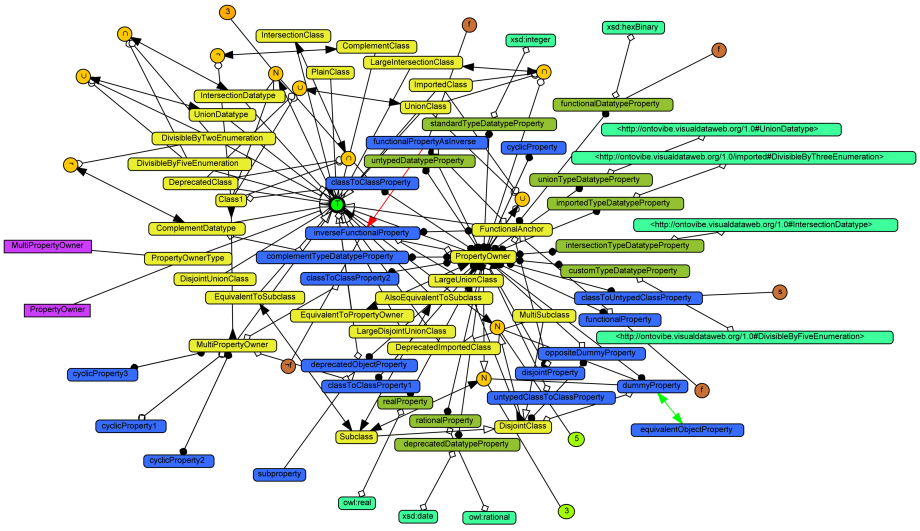


Fig. 1. OntoViBe visualized with SOVA.

SOVA is a plugin for Protégé that provides graph visualizations of ontologies [5]. When displaying OntoViBe in SOVA 0.8.4 (Figure 1), the distinction of classes, object and datatype properties is instantly visible—though relying purely on colors. Based on OntoViBe, support for functional, inverse functional, and symmetric properties can be seen, as they are marked by little brown circles with short abbreviations for the property characteristics.

Furthermore, some of the limitations of the SOVA implementation can be identified. `PropertyOwner` and `MultiPropertyOwner` are classes, but at the same time, they are instances of the class `PropertyOwnerType`. SOVA displays them twice, once as classes and once as individuals, rather than as a single concept. Cardinality constraints are displayed as small colored circles, which are easy to spot—for a combined minimum and maximum cardinality constraint, however, only the lower bound (the number 5 in the green circle) is shown. Overall, the visualization contains many edges and edge crossings, which significantly reduce its readability. A large number of these edges result from the fact that all implicit subclass relations to `owl:Thing` are depicted in the SOVA visualization, and that every piece of information is shown in a separate node.

4.2 VOWL

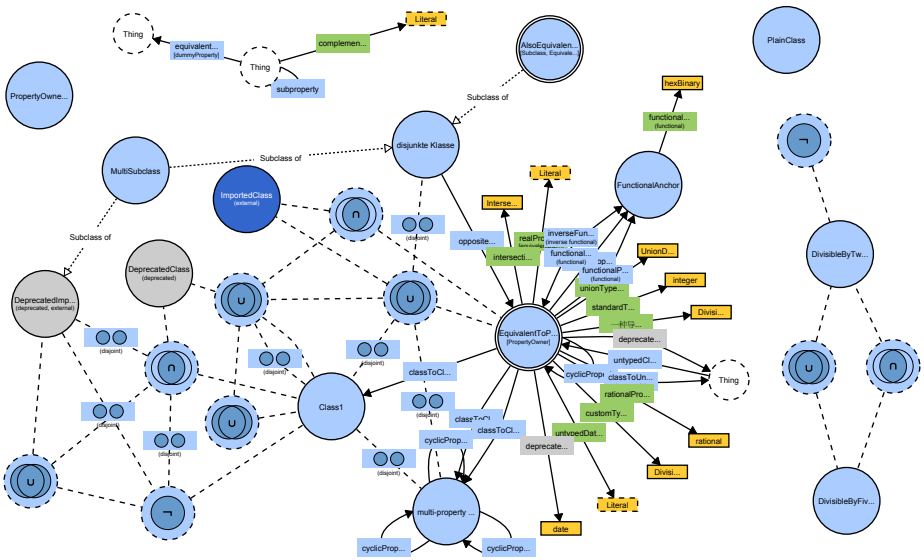


Fig. 2. OntoViBe visualized with VOWL.

VOWL, the Visual Notation for OWL Ontologies, was developed as a means to both obtain a structural overview of OWL ontologies and recognize various attributes of ontology elements at a glance [11]. It has been implemented

in two different tools, a plugin for Protégé and a responsive web application. Figure 2 has been created with version 0.2.15 of the web application (called WebVOWL [10]) that is available at `http://vowl.visualdataweb.org`.

Applying VOWL to OntoViBe shows some typical characteristics of VOWL visualizations, such as equivalent classes being represented as one class with a double border, other special elements being multiplied in the visualization (e.g., `owl:Thing`), or text in brackets below the labels indicating attributes such as *functional* or *symmetric*. Like in SOVA, custom data ranges are not (yet) completely shown, as is seen by the respective nodes that simply display the names of the data ranges (e.g., "Divisib..." for "DivisibleByFiveEnumeration") but no more information on how they are defined. Moreover, it becomes apparent that the WebVOWL implementation tends to route edges between the same pairs of nodes in a way so as to avoid overlapping labels. As the implicit subclass relations to `owl:Thing` are not shown in VOWL, the nodes of the graph visualization are less connected than in SOVA.

4.3 OWLViz

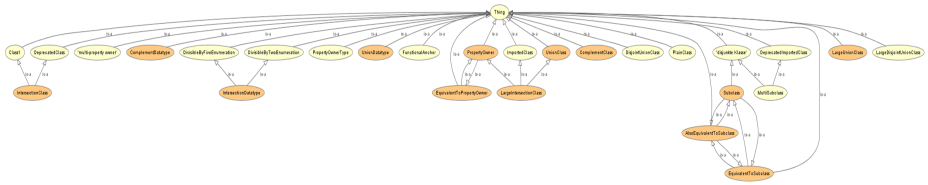


Fig. 3. OntoViBe visualized with OWLViz.

The OWLViz visualization is aimed at visualizing exclusively the hierarchical class structure of ontologies. When used to visualize OntoViBe (Figure 3), the fact that only inheritance ("is-a") relationships are shown by OWLViz 4.1.2, gets apparent. Also, it gets clear that equivalence relationships between classes are expressed as bidirectional inheritance. Other property relations are not visualized in OWLViz, and also further class or property characteristics are not included, making OWLViz an ontology visualization with very limited expressiveness.

4.4 OntoGraf

OntoGraf (Figure 4) depicts property relations between classes with colored lines. Given that OntoViBe includes properties of different types and with different characteristics, it is notable that these are not displayed by OntoGraf 1.0.1 in an inherently distinct way. Again, the graph is highly connected, as the implicit subclass relations to `owl:Thing` are explicitly shown.

Like in OWLViz (Section 4.3), equivalence between classes is displayed by two opposite inheritance arrows. Additionally, classes that are equivalent to others

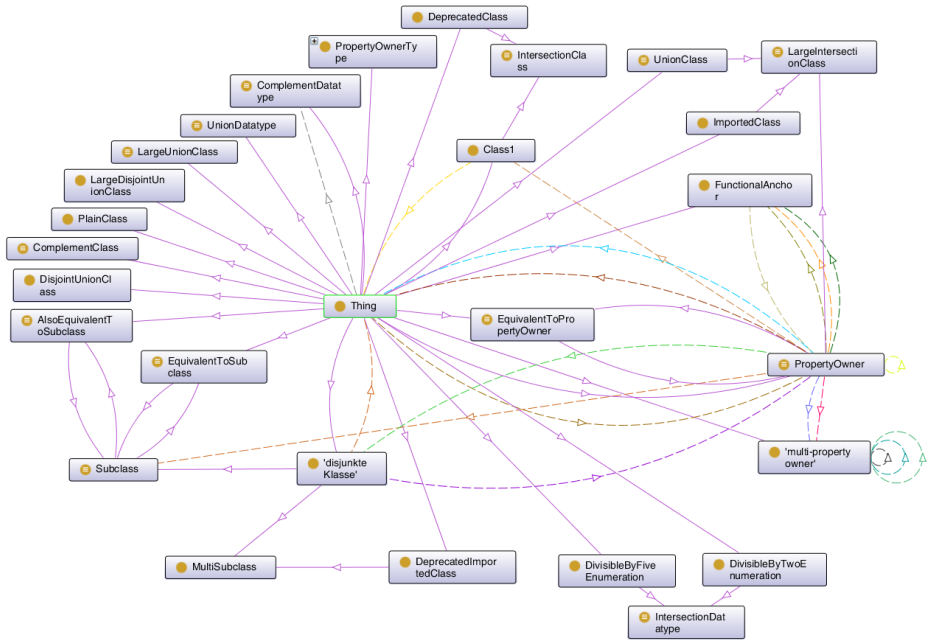


Fig. 4. OntoViBe visualized with OntoGraf.

are highlighted by an equivalence symbol. Other than that, the test shows that OntoGraf copes well with several cyclic properties applied to the same class.

4.5 LODE

LODE is a documentation generator for ontologies [14]. While LODE is not a visualization approach, the output of version 1.2 after processing OntoViBe can be examined in a similar fashion. Features that get apparent in the excerpt shown in Figure 5 include the transformation of the camel-cased element names into separate words (e.g., `DeprecatedClass` becomes *deprecated class*), and the lists of superclasses, subclasses, and connected properties per class.

5 Conclusion and Future Work

Based on the OWL 2 specifications, we have defined OntoViBe, a benchmark ontology for testing ontology visualizations. We did not focus on scalability or other common benchmark goals, such as execution speed, but rather on feature completeness and flexibility in terms of combination of elements with regard to the OWL 2 specifications. Since OWL may further evolve in the future, OntoViBe needs to keep being updated accordingly.

Features not included in OntoViBe may be considered for future adjuncts of the ontology. For instance, these could be separate modules that focus on

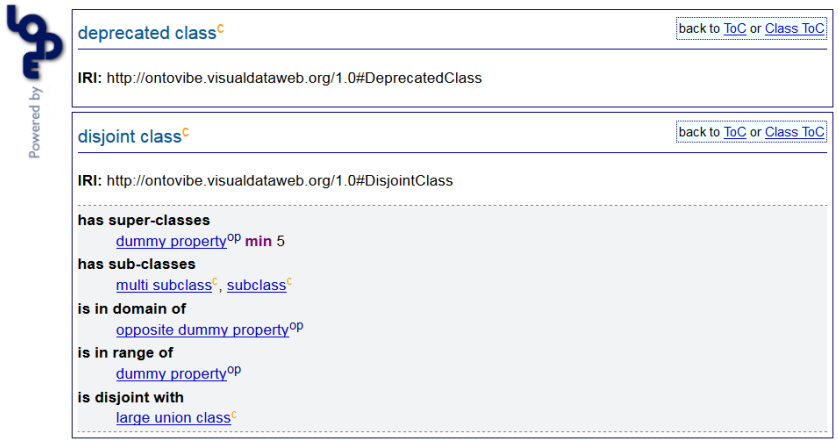


Fig. 5. Excerpt of the HTML documentation for OntoViBe generated by LODE.

testing specific aspects, such as combinations of cardinality constraints or the population of OntoViBe with individuals and other ABox information.

Generally, we hope that our experiences from the development of OntoViBe can benefit other projects, including benchmark data models beyond the task of ontology visualization.

References

1. Ontology alignment evaluation initiative. `http://oaei.ontologymatching.org`
2. Protégé ontology editor. `http://protege.stanford.edu`
3. The OWL 2 schema vocabulary (OWL 2). `http://www.w3.org/2002/07/owl.rdf` (2009)
4. Bail, S., Parsia, B., Sattler, U.: JustBench: A framework for OWL benchmarking. In: Proceedings of the 9th International Semantic Web Conference (ISWC '10), pp. 32–47. Springer (2010)
5. Boinski, T., Jaworska, A., Kleczkowski, R., Kunowski, P.: Ontology visualization. In: Proceedings of the 2nd International Conference on Information Technology (ICIT '10). pp. 17–20. IEEE (2010)
6. Carroll, J.J., Roo, J.D.: OWL web ontology language test cases. `http://www.w3.org/TR/owl-test/` (2004)
7. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14), pp. 137–152. Springer (2014)
8. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Web Semantics 3(2–3), 158–182 (2005)
9. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E.: Ontology visualization methods – a survey. ACM Computing Surveys 39(4), 10:1–10:43 (2007)

10. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: Proceedings of EKAW 2014 Satellite Events. Springer (to appear)
11. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: User-oriented visualization of ontologies. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14), pp. 266–281. Springer (2014)
12. Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Proceedings of the 3rd European Semantic Web Conference (ESWC '06), pp. 125–139. Springer (2006)
13. Meilicke, C., García-Castro, R., Freitas, F., Van Hage, W.R., Montiel-Ponsoda, E., Ribeiro De Azevedo, R., Stuckenschmidt, H., Šváb Zamazal, O., Svátek, V., Tamilin, A., Trojahn, C., Wang, S.: MultiFarm: A benchmark for multilingual ontology matching. Web Semantics 15, 62–68 (2012)
14. Peroni, S., Shotton, D., Vitali, F.: The live OWL documentation environment: A tool for the automatic generation of ontology documentation. In: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management (EKAW '12). pp. 398–412. Springer (2012)
15. Smith, M., Horrocks, I., Krtzsch, M., Glimm, B.: OWL 2 web ontology language conformance (second edition). `http://www.w3.org/TR/owl2-conformance/` (2012)
16. Šváb, O., Svátek, V., Berka, P., Rak, D., Tomášek, P.: OntoFarm: Towards an experimental collection of parallel ontologies. In: Poster Track of ISWC 2005 (2005)
17. W3C OWL Working Group: OWL 2 web ontology language document overview (second edition). `http://www.w3.org/TR/owl2-overview/` (2012)
18. Wang, S.Y., Guo, Y., Qasem, A., Heflin, J.: Rapid benchmarking for semantic web knowledge base systems. In: Proceedings of the 4th International Semantic Web Conference (ISWC '06), pp. 758–772. Springer (2005)

Table 1. Coverage table of OntoViBe elements with respect to OWL 2 features.

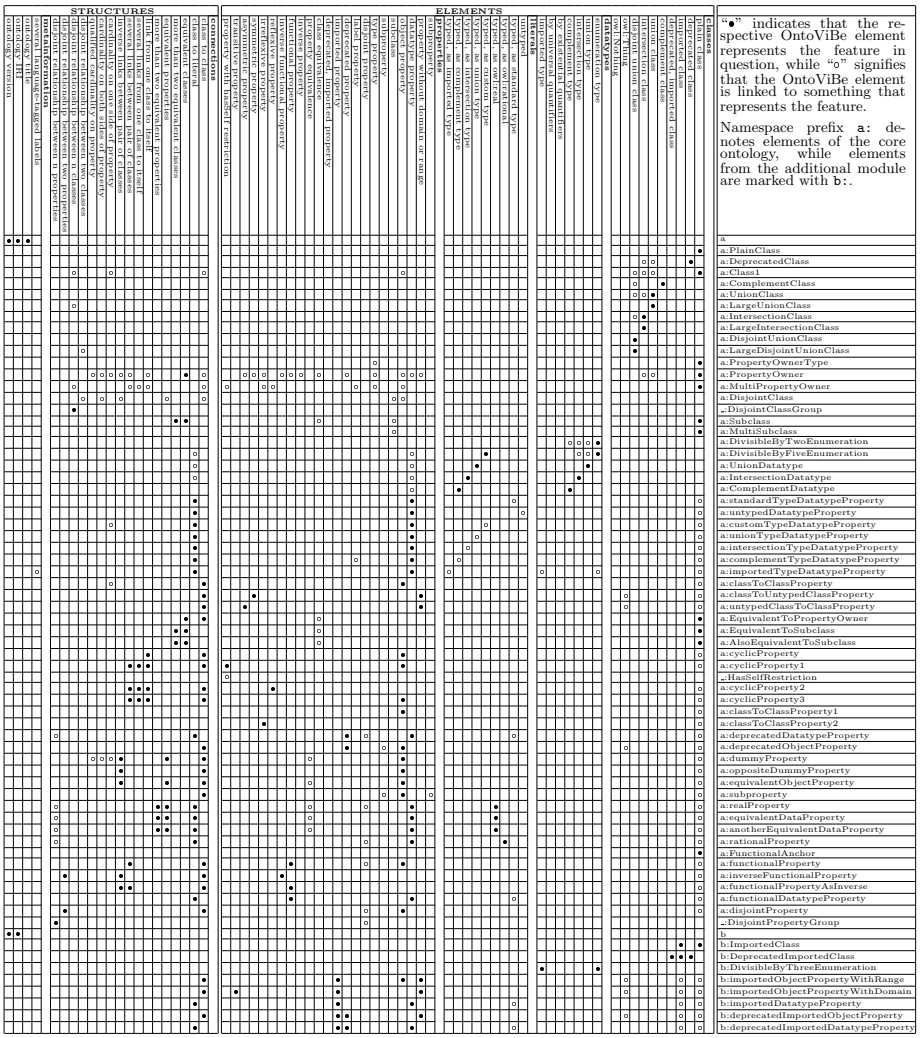
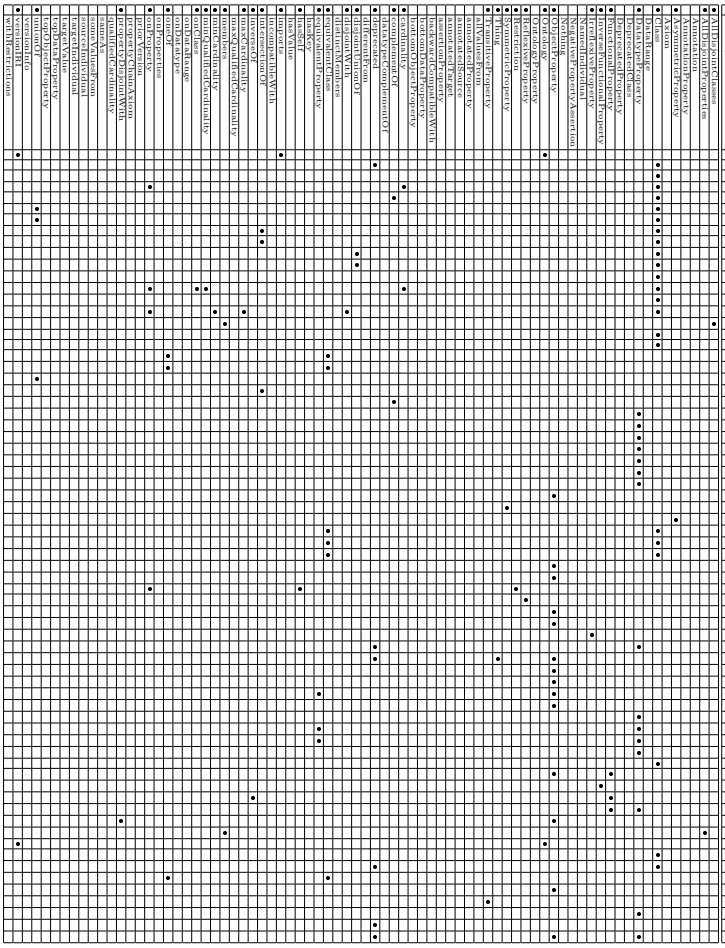


Table 2. Coverage table of OntoViBe elements with respect to OWL 2 identifiers.



What Can the Ontology Describe? Visualizing Local Coverage in PURO Modeler

Marek Dudáš, Tomáš Hanzal, and Vojtěch Svátek

University of Economics, Prague,
marek.dudas|xhant00|svatek@vse.cz

Abstract. Ontologies and vocabularies written in OWL are a crucial part of the semantic web. OWL however allows to model the same part of reality using different combinations of constructs, constituting 'modeling styles'. Comparing how different ontologies from a similar domain cover a specific part of reality might be more difficult when each ontology uses a different style. PURO, a language for ontological background models, can serve as mediator, as it allows to create models that are directly mappable to OWL but overcome its unnatural modeling limits dictated by description logic. By highlighting the parts of the PURO model covered by particular ontologies, the ontology coverage comparison can be visualized. We demonstrate this approach using a simple graphical tool.

1 Introduction

OWL [8], the language of ontologies on the semantic web, allows to model the same real-world state of affairs using different combinations of language constructs, a kind of modeling styles. For example, in a lightweight ontology, data properties might be preferred, while in a more complex ontology, object properties might be used for describing the same relationships, allowing to state more facts about the property value. Current ontology visualization tools (as surveyed in [4,6]) either display the OWL constructs directly or only offer very lightweight generalization, as in [5]. When the user wants to compare, in a visual manner, the *local coverage*[1] of different ontologies, i.e. their capability to express a certain cluster of relationships, s/he has to first translate the OWL language constructs into his/her mental model to abstract from the modeling differences.

A possible solution in such a situation is to express the above mentioned cluster of relationships in a modeling language that allows to abstract from the modeling style differences and thus make the mental model explicit. We believe that *PURO ontological background models* (OBMs) [9] might fit this use case. We are developing PURO Modeler, a Javascript-based tool for graphical design of OBMs and ontology local coverage comparison.[2] In the paper, we demonstrate its usage on two examples.

[1] We assume that typically only a few entities and relationships are considered from the analyzed models (e.g., a book, its author and its subject). To differ from the coverage of the whole domain (e.g., bibliography), we use the term 'local coverage'.

[2] The development version is available at `http://protegeserver.cz/puromodeler`.

Related Research We are unaware of any prior research on visual comparison of ontology local coverage. Creating and visualizing mappings [3] between the compared ontologies might be useful but it only shows what the ontologies have in common. There is research on ontology comparison (e.g., [7]), but that is rather focused on automatic computation of ontology similarity. In our approach the actual analysis is left to the user; we however propose a method and means for supportive visualization. The PURO language, designed to be easily mappable to OWL, is used as interlingua for this purpose. Other modeling languages, e.g., entity-relationship diagrams [2], might be considered as alternatives to PURO. Of those, *OntoUML* [1], a version of UML for conceptual modeling, is probably the closest match; an existing tool, OLED, even allows to transform OntoUML models into OWL fragments. However, the purpose of OntoUML is conceptual modeling with easy validation and it is not designed for OWL ontology development or usage.

2 PURO Ontological Background Models

The long-term vision of the PURO OBM language is to model the real world, as much as possible, 'as it is', while remaining 'very close' to OWL. It is only intended to serve as an aid in ontological engineering, i.e. models created in it are not supposed to serve (in large scale) as schema for data or input to reasoners. The liberation from the constraints dictated by description logic allowed to drop some constraints of OWL (DL, or even Full) that are often perceived as unnatural; most notably, meta-concepts (such as classes of classes, or classes as 'property values') and arbitrary n-ary relationships can be expressed in PURO.

PURO OBMs are based on two main distinctions: between particulars and universals and between relationships and objects (hence the PURO acronym: Particular-Universal, Relationship-Object). In simple terms: universals, i.e. types, can have instances, while particulars cannot (P-U distinction); objects and relationships are differed by their identification: objects are entities with their own identity that can be 'talked about' independently, while talking about relationships always 'brings in' their participating entities (R-O distinction). There are six basic terms: B-object (particular object), B-type (type of objects/types), B-relationship (particular relationship), B-relation (type of relationship), B-valuation (particular assertion of quantitative value) and B-attribute (type of valuation).

An OBM consists of named *instances* of these terms and relationships between them.[3] To say, it captures a concrete situation that serves as basis for either analyzing existing ontologies or generating new ones.

3 Using PURO Modeler for Local Coverage Comparison

PURO Modeler is basically a graph editor. The user can create an OBM where the terms are represented by nodes and the relationships between them by edges.

[3] The description of the language is simplified, more details are in [10].

As the OBMs are models of specific real-world situations, such an example cluster of relationships has to be chosen. After constructing the model,[4] or loading an already existing one (possibly created by another user and shared), the user can select each node and annotate it with the ontologies that this particular instance of PURO term is covered by (i.e., the instance can be described with that ontology). The parts of the graph locally covered by each vocabulary are then highlighted so that the coverage could be easily compared in one diagram.

3.1 Example 1: A Book and its Issue

Let us say we need to annotate data about books and their various issues (i.e. their manifestations). We can find several ontologies that might be used for it. **BIBFRAME**[5] contains class Text as a subclass of *Work*, class Print as a subclass of *Instance* and the *hasInstance* property for stating that an instance of *Print* is a manifestation of some *Text*. Figure 1b shows an example of possible usage of BIBFRAME to describe the book *The Semantic Web for the Working Ontologist* and its printed paperback manifestation. **FRBR** ontology[6] (and its extension **Fabio**[7]) distinguishes between four different 'levels of abstraction' represented by classes *Work*, *Expression*, *Manifestation* and *Item*, each with subclasses representing more specific types. There is a set of properties that can be used to link the instances of the classes, as shown in Figure 1c. **Schema.org** does not distinguish between the work and its manifestation through class membership, but contains properties *exampleOfWork* and *workExample* that serves that purpose. It contains the class *Book* and allows to specify the format of its instances by linking them to instances of *BookFormatType* with property *bookFormat* (Figure 1a).

For the purposes of the local coverage comparison, we chose the book *The Semantic Web for the Working Ontologist* (as an instance of the B-type Text, which is a subtype of Work) and its 2009 paperback issue (an instance of the B-type Paperback, a subtype of Book). An OBM of these two entities accompanied by a B-object representing an exemplar of the paperback issue (as a tangible item) and the relationships between them is in Figure 2 (exported as a screenshot from PURO Modeler). The color-coded highlighted parts of the model shows what of it can be expressed in each vocabulary.

We can see that, e.g., the physical exemplar of the issue can only be described with FRBR (which 'implements' the whole OBM), BIBFRAME allows to describe an issue of a book, but without specifying its format, and Schema.org does not allow to say that an instance of work is a *Text*.

[4] The methodology for the OBM modeling is yet to be formalized.
[5] `http://bibframe.org/vocab`
[6] `http://purl.org/vocab/frbr/core#`
[7] `http://purl.org/spar/fabio/`

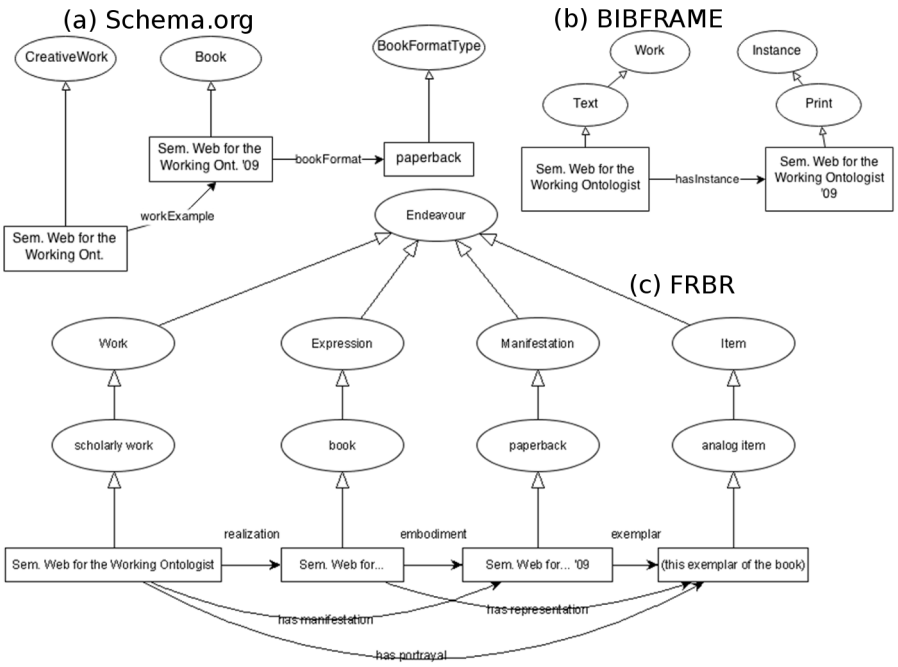


Fig. 1. Diagrams showing three different OWL modeling styles of the relationship between a book and its issue.

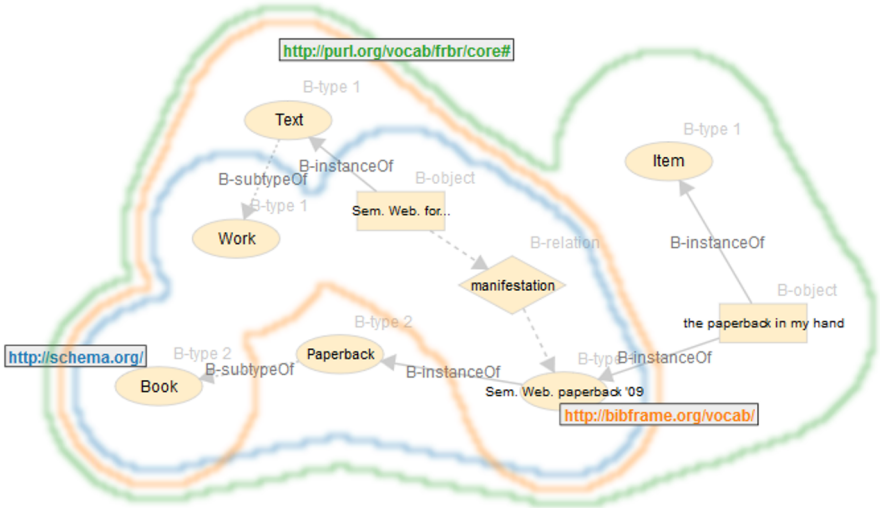


Fig. 2. The OBM of the relationship between a book and its issue. The amount of its coverage in various OWL representations (shown in Figure 1) is highlighted.

3.2 Example 2: A Dish and a Recipe

In the second example, we compare the local coverage of the relationships between ingredients, a recipe and a dish produced by it. We analyzed[8] three ontologies: Schema.org, Food Ontology[9] and Linked Recipes.[10] Their local coverage is highlighted in Figure 3 (for simplicity, we made up a 'boiled egg recipe' with only one ingredient and we did not include all possible B-attributes of the B-objects, like 'protein content' of the dish).

We can see that Schema.org allows to describe the recipe in terms of its origin, ingredients and time needed; the carbohydrates, fat and energy content; and size of the produced food. It does not allow to model the recipe instructions: there is only one datatype property for entering a free text description. Linked Recipes does not allow to describe the characteristics of the recipe product, but (unlike Schema.org) allows to type it as 'Food'. It can be used to annotate instructions including their order, and ingredients including their quantity. Food Ontology can only be used for the (quite detailed) description of the ingredients and the resulting dish. It does not contain classes/properties relevant for cooking instructions. None of the three ontologies covers the whole situation as modeled by the OBM, however, we can see that a combination of two of them might be sufficient, e.g., using Linked Recipes for recipes in combination with Food Ontology for more detailed ingredient and dish description.

4 Conclusions and Future Work

We argued that the task of ontology comparison in terms of local coverage is difficult if the ontologies use different modeling styles. We proposed that using a more general modeling language for visualizing the local coverage of several ontologies in one place, abstracting from the OWL modeling differences, might make the comparison easier, and that the PURO OBM language might be suitable for such a use case. As a first step towards the evaluation of the approach we implemented PURO Modeler: a Javascript application for PURO OBM models design and visualization of the amount of their local coverage in various ontologies. We demonstrated the usage of PURO Modeler for local coverage comparison on two examples, each consisting of an analysis of three ontologies.

The future work might include, besides the full evaluation, building a portal where the developers could publish visualizations of their ontology/vocabulary local coverage of typical situations. Such a website could then serve as an addition or enhancement of the existing Linked Open Vocabularies[11] portal. Intense ongoing research also addresses the obvious bottleneck of the approach: the design of plausible PURO OBMs. It is going to be supported by interactive guidelines and NLP-based model verbalization widgets.

This research is supported by the VŠE IGA project no. F4/34/2014.

[8] See `http://tomhanzal.github.io/owl-modeling-styles/#part2` for full analysis.
[9] `http://data.lirmm.fr/ontologies/food/`
[10] `http://linkedrecipes.org/schema/`
[11] `http://lov.okfn.org/dataset/lov/`

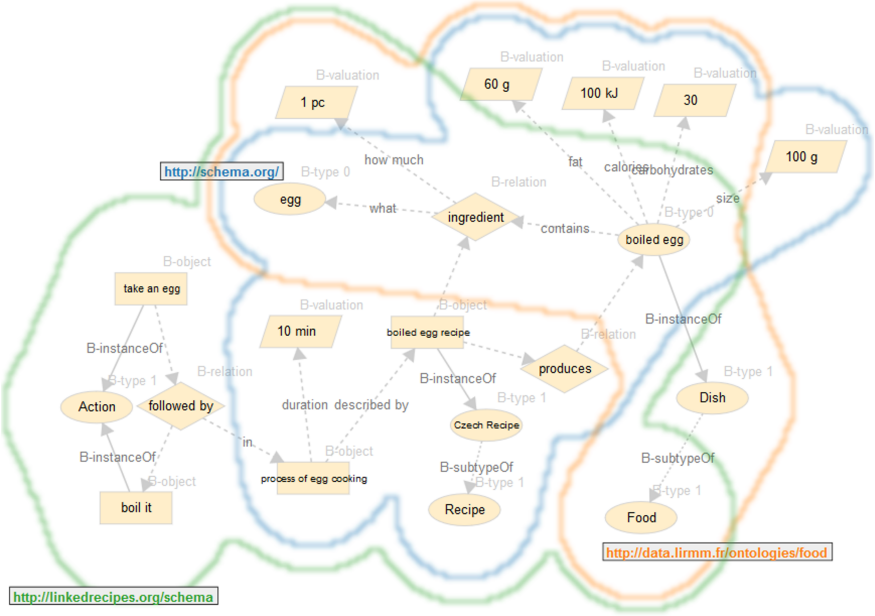


Fig. 3. The OBM of a recipe, ingredients and dish. The amount of its coverage in various OWL representations is highlighted.

References

1. Albuquerque, A., Guizzardi, G.: An ontological foundation for conceptual modeling datatypes based on semantic reference spaces. In Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference, 2013.
2. Chen, P.: The entity-relationship model – toward a unified view of data. In: *ACM Transactions on Database Systems* (TODS), 1976, 1.1: 9-36.
3. Choi, N., Song, I. Y., Han, H.: A survey on ontology mapping. ACM Sigmod Record, 35(3), 34-41. 2006.
4. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and Navigating in the Ontology Visualization Landscape. Accepted to EKAW 2014.
5. Hayes, P., et al.: Collaborative knowledge capture in ontologies. In: Proceedings of the 3rd international conference on Knowledge capture – K-CAP 2005, ACM, 2005.
6. Katifori, A., et al.: Ontology Visualization Methods – A Survey. *ACM Comput. Surv.* 39, 10143. 2007.
7. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Knowledge engineering and knowledge management: Ontologies and the semantic web. Springer Berlin Heidelberg, 2002. p. 251-263.
8. *OWL 2 Web Ontology Language Document Overview.* W3C Recommendation, 2009.
9. Svátek, V., et al.: Mapping Structural Design Patterns in OWL to Ontological Background Models. In: *K-CAP 2013*, ACM, 2013.
10. Svátek, V., et al.: Metamodeling-Based Coherence Checking of OWL Vocabulary Background Models. In: *OWLED 2013*, online `http://ceur-ws.org/Vol-1080/owled2013_6.pdf`.

User Involvement for Large-Scale Ontology Alignment

Valentina Ivanova and Patrick Lambrix

Department of Computer and Information Science and the Swedish e-Science Research Centre
Linköping University, 581 83 Linköping, Sweden

Abstract. Currently one of the challenges for the ontology alignment community is the user involvement in the alignment process. At the same time, the focus of the community has shifted towards large-scale matching which introduces an additional dimension to this issue. This paper aims to provide a set of requirements that foster the user involvement for large-scale ontology alignment tasks and a state of the art overview.

1 Motivation

The growth of the ontology alignment area in the past ten years has led to the development of a number of ontology alignment tools. The progress in the field has been accelerated by an annual evaluation initiative (Ontology Alignment Evaluation Initiative—OAEI) which has provided a discussion forum for developers and a platform for an annual evaluation of their tools. The number of systems participating in the evaluation increases each year, yet few provide a user interface and even fewer navigational aids or complex visualization techniques. Some systems provide scalable ontology alignment algorithms, however, for achieving high-quality alignments user involvement during the process is indispensable.

Nearly half of the challenges identified in [24] are directly related to user involvement. These include *explanation of matching results* to users, fostering the *user involvement* in the matching process and *social and collaborative matching*. Another challenge aims at supporting users' collaboration by providing *infrastructure and support* during all phases of the alignment process. All these challenges can be addressed by providing user interfaces in combination with suitable visualization techniques.

The demand for user involvement has been recognized by the alignment community and resulted in the introduction of the OAEI Interactive track in 2013. Quality measures for evaluation of interactive ontology alignment tools have been proposed in [20]. The results from the first edition of the track [3] show the benefits from introducing user interactions (in comparison with the systems' non-interactive modes) by means of increasing the precision for all (five) participants and the recall for three of them. The test cases presented in [9] show that simulating user interactions with 30% error rate during the alignment process has led to the same results as a non-interactive matching.

With the development of the ontology engineering field the size and complexity of the ontologies, the alignments and, consequently, the matching problems increase as emphasized in [24] by the *large-scale matching evaluation* challenge. This trend is demanding scalable and (perhaps) novel user interfaces and interactions which is going to impose even stricter scalability requirements towards the algorithms in order to provide

timely response to the users. For instance, graph drawing algorithms should not introduce delays in order for a tool to provide interactive visualization. Scalability, not only in terms of computation, but also in terms of interaction is one of the crucial features for the ontology alignment systems according to [9]. According to [22] user interactions are essential (in the context of large ontologies) for configuring the matching process, incremental matching and providing feedback to the system regarding the generated mapping suggestions.

Currently the alignment systems focus on their main task—ontology alignment—with little or no support for an infrastructure or functionalities which are not directly related to the alignment process. Coping with the increasing size and complexity of ontologies and alignments will require not only comprehensive visualization and user interactions but also supporting functionalities not directly related to them as discussed in subsection 2.2. For instance, the authors in [6] identify cognitive support requirements for alignment tools not directly related to the alignment process—interrupting/resuming the alignment process and providing a feedback on its state. Achieving collaborative matching, discussed above, is going to need a suitable environment.

This paper aims to provide requirements for ontology alignment tools that encourage user involvement for large-scale ontology alignment tasks (section 2). Several ontology alignment systems are evaluated in section 3 in connection with the requirements in section 2. Section 4 provides a discussion and section 5 concludes the paper.

2 Requirements for User Support in Large-Scale Ontology Alignment

This section presents requirements for ontology alignment systems meant to foster user engagement for large-scale ontology alignment problems. Subsection 2.1 summarizes the requirements presented in [6] which address the cognitive support that should be provided by an alignment system to a user during the alignment process. While they are essential for every alignment system, the focus in the community has shifted towards large-scale matching since the time they have been developed. Thus other requirements to assist the user in managing larger and more complex ontologies and alignments are in demand (subsection 2.2). They may not always be directly related to visualization and user interactions but contribute to the development of a complete infrastructure that supports the users during large-scale alignment tasks. Those requirements are extracted from existing works and systems and from the authors' personal experience from the development of ontology alignment and debugging systems ([13], [12]). Since those requirements address user involvement as well they sometimes overlap with those in subsection 2.1 and can be considered complementary to them.

The requirements discussed in this section are crucial for large-scale alignment tasks, but also beneficial for aligning small and medium size ontologies. While the alignment of two medium size ontologies is feasible on a single occasion by a single user even without techniques for reducing user interventions, the alignment of large-scale ontologies without such techniques would be infeasible.

The authors in [7] identify requirements for supporting user interactions in alignment systems which can be seen as a subset of those in subsections 2.1 and 2.2. The

Dimensions	Requirements
Analysis and Generation Dimension	#3.1: automatic discovery of some mappings; #3.2: test mappings by automatically transforming instances between ontologies; #3.3: support potential interruptions by saving and returning users to given state; #3.4: support identification and guidance for resolving conflicts;
Representation Dimension	#4.1: visual representation of the source and target ontology; (I) #4.2: representation of a potential mapping describing why it was suggested, where the terms are in the ontologies, and their context; (I,E) #4.3: representation of the verified mappings that describe why the mapping was accepted, where the terms are in the ontologies, and their context; (I,E) #4.4: identify visually candidate-heavy regions; (I) #4.5: indicate possible start points for the user; (E) #4.6: progress feedback on the overall mapping process; (E) #4.7: feedback explaining how the tool determined a potential mapping; (E)
Analysis and Decision Making Dimension	#1.1: ontology exploration and manual creation of mappings; (I,M) tooling for the creation of temporary mappings; (M) #1.2: method for the user to accept/reject a suggested mapping; (M) #1.3: access to full definitions of ontology terms; (I) #1.4: show the context of a term when a user is inspecting a suggestion; (I)
Interaction Dimension	#2.1: interactive access to source and target ontologies; (I) #2.2: interactive navigation and allow the user to accept/reject suggestions; (I,M) #2.3: interactive navigation and removal of verified mappings; (I,M) #2.4: searching and filtering the ontologies and mappings; (I) #2.5: adding details on verified mappings and manually create mappings; (M)

Table 1. Cognitive support requirements adapted from [6].

same applies for those in [5] which lists requirements for alignment editors and visualizers relevant for individual and collaborative matching and explanation of the alignments.

2.1 Cognitive Support Requirements

The requirements identified in [6] are based on research in the area of cognitive theories and a small user study with four participants. They are grouped in four conceptual dimensions (table 1).

The *Analysis and Generation* dimension includes functions for automatic computation and trial execution of mapping suggestions (potential mappings), inconsistency detection/resolution and services for interrupting/resuming the alignment process. The mappings and mapping suggestions together with explanations why/how they are suggested/accepted are visualized by services in the *Representation* dimension. Other functions include interactions for overview and exploration of the ontologies and alignments and feedback for the state of the process. Requirements 1, 2 and 3 from [7] and the first and the third requirements in [5] focus on similar services. The *Analysis and Decision Making* dimension considers the users' internal decision making processes and involves exploration of the ontology terms and their context during the process of discovering

and creating (temporary) mappings, and validating mapping suggestions. During the *Interaction* dimension the user interacts with the system through its exploration, filtering and searching services in order to materialize his/her decisions by creating mappings and accepting/rejecting mapping suggestions. Requirements 4, 5 and 6 from [7] cover similar interactions. Such requirements are also identified in [5]. The requirements for the *Analysis and Decision Making* dimension can be considered to utilize the functionalities represented by the requirements in the *Interaction* dimension.

The requirements provided by the *Representation* and *Interaction* dimensions are involved in the human-system interaction and can be roughly separated in the following three categories—manipulation (M), inspection (I) and explanatory (E) requirements. Those in the first category include actions for transforming the mapping suggestions in an alignment—accept/reject mapping suggestions, add metadata and manually create mappings, etc. Similar functionalities are needed for the ontologies (#5.0), as well, since the user may need to, for instance, introduce a concept in order to provide more accurate mappings, as described in [16] as well. Those in the second category cover a broad set of actions for inspecting the ontologies and alignments—exploring the ontologies, mappings and mapping suggestions, search and filter by various criteria, zoom, overview, etc. The third category includes services for presenting information to the user, for instance, reasons to suggest/accept a mapping suggestion, how the tool has calculated it, hinting at possible starting points and showing the current state of the process.

2.2 Ontology Alignment in Large Scale

Various requirements arise from the tendency of increasing the size and complexity of the ontologies, alignments and alignment problems. They need to be supported by scalable visualization and interaction techniques as well. For instance, an introduction of a debugging phase during the alignment process (discussed below) will demand adequate presentation of the defects and their causes which is a problem of the same scale as the main problem discussed in this paper. This subsection does not discuss the techniques for large-scale matching identified in [22] or matching with background knowledge since they are not directly related to user involvement. However some of those techniques affect the interactivity of the systems and thus indirectly influence the user involvement.

Aligning large and complex ontologies cannot be handled on a single occasion. Thus the user should be able to suspend the process, preserve its state and resume it at another point in time (#3.3). Such **interruptions of the alignment process (#5.1)** may take place during different stages, for instance, during the computation of mapping suggestions, during their validation, etc. At the time of interruption the system may provide partial results which can be reused when the alignment process has been resumed. The SAMBO system [13] implements such approach introducing interruptible computation, validation and recommendation sessions. Requirement 9 in [7] can be seen as similar, but without saving and reusing already validated suggestions.

Another strategy to deal with large-scale tasks is to **divide** them **into smaller tasks (#5.2)**. This can be achieved by clustering algorithms or grouping heuristics. Smaller problems can be more easily managed by single users and devices with limited resources. Requirement 8 from [7] proposes distributing parts of the task among several

users. The authors of AlViz [15] highlight that clustering the graph improves the interactivity of the program (by reducing the size of the problem). Clustering of the ontologies and alignments will allow reusing visualization techniques that work for smaller problems. A fragment-based strategy is implemented in [4] where the authors also note that not all fragments in one schema would have corresponding fragments in another.

In the context of large-scale matching it is not feasible for a user to validate all mapping suggestions generated by a system, i.e., tools' developers should aim at **reducing unnecessary user interventions (#5.3)**. The authors in [20] define a measure for evaluating interactive matching tools based on the number and type of user interventions in connection with the achieved F-measure. LogMap2 [9] only requires user validation for problematic suggestions. In [13] the authors demonstrate that the session-based approach can reduce the unnecessary user interventions by utilizing the knowledge from previously validated suggestions. GOMMA [11] can reuse mappings between older ontology versions in order to match their newer versions. PROMPT [17] logs the operations performed for merging/aligning two ontologies and can automatically reapply them if needed. Reducing the user interventions, but at the same time effectively combining manual validation with automatic computations are two of the challenges identified in [19]. The authors in [2] and [23] discuss criteria for selecting mapping suggestions that are shown to the user and strategies for user feedback propagation in order to reduce the user-system interactions.

Matching large ontologies is a lengthy and demanding task for a single user. It can be relaxed by involving several users who can discuss together and decide on problematic mappings in a collaborative environment. The **social and collaborative matching (#5.4)** is still a challenge for the alignment community [24]. Requirement 7 in [7] addresses this open opportunity. It has potential to reduce the load of a single user and the number of incorrect mappings by building on the collective knowledge of a number of people who can review mappings created by other participants [5]. One of the quality aspects for ontology alignment discussed in [16] is the social aspect—it can be achieved by means of collaboration and information visualization techniques.

Another challenge insufficiently addressed [24] by the alignment community is related to the **environment (#5.5)** where such collaboration could happen. Apart from aligning ontologies it should also support a variety of functions for managing alignments such as storing/editing/retrieving/sharing alignments as explained in [5]. Accommodating different versions of alignments, for instance, would require an entire infrastructure on its own and probably a permanent storage similarly to GOMMA/COMA++. The environment should support services for communication between its members like discussion lists, wikis, subscriptions/notifications, messages, annotations, etc.

Providing **recommendations (#5.6)** is another approach to support the user during the decision making process. Such recommendations can be based on external resources, previous user actions, based on other users' actions (in a collaborative environment), etc. They can be present at each point user intervention is needed—choosing an initial matcher configuration [1], validating mapping suggestions [12], choosing a starting point, etc. The authors in [13] implement recommendation sessions which match small parts of the selected ontologies in order to recommend the best settings for matching them. Different weights can be assigned to the recommendations depending on their

sources. Suitable ranking/sorting strategies could be applied to present them in a particular order.

The outcome of the applications that consume alignments is directly dependent on the quality of the alignments. A direct step towards improving the quality of the alignments and, consequently, the results from such applications is an introduction of **a debugging step during the alignment process (#5.7)**. It was shown in [8] that a domain expert has changed his decisions regarding mappings he had manually created, after an interaction with a debugging system. Most of the alignments produced in the Anatomy, LargeBio and even Conference (which deals with medium size ontologies) tracks in OAEI 2013 [3] are incoherent which questions the quality of the results of the semantically-enabled applications utilizing them. According to [9] *reasoning-based error diagnosis* is one of the three essential features for alignment systems. Almost half of the quality aspects for ontology alignment defined in [16] address lack of correctness in the alignment in terms of *syntactic*, *semantic* and *taxonomic* aspects. The trends toward increasing the size and complexity of the alignment problem demand debugging techniques more than ever. In this context a debugging module should be present in every alignment system. The authors in [10] show that repairing alignments is feasible at runtime and improves their logical coherence when (approximate) mapping repairing techniques are applied. Since ontology debugging presents considerable cognitive complexity (due to the, potentially, long chains of entailments) adequate visual support to aid user interactions is a necessity.

In the field of ontology debugging there is already ongoing work that addresses explanation of defects to users. These techniques could be borrowed and applied in the ontology alignment to address the challenge for **explaining the matching results** to the users (#4.2, #4.7). The authors in [19] specify generating human understandable explanations for the mappings as a challenge as well. The authors in [1] implement advanced interfaces for **configuring** the **matching process (#5.8)** which provide the users with insights of the process and contribute to the understanding of the matching results.

Trial execution of mappings (#5.9.1) (what-if) mentioned above in the context of confirming user's expectations (#3.2) will be of even greater help during the debugging and alignment by aiding the user in the propagation of the consequences of his/her actions. Additionally **support for temporary decisions (#5.9.2)** in general, including temporary mappings (#1.1), list of performed actions and undo/redo actions, will help the user to explore the effects of his/her actions (and reduce the cognitive load).

3 Overview of Ontology Alignment Systems

The systems in this literature study are selected because they have mature interfaces, often appear in user interface evaluations and accommodate features addressing the alignment of large ontologies.

3.1 AlViz

AlViz [15] is a Protégé plug-in which uses the linking and brushing paradigm for connecting multiple views of the same data where navigation in one of the views changes

	Requirements	AlViz	SAMBO	PROMPT	CogZ	RepOSE	AML	COMA
manipulate	#2.5;1.1 create mapping manually	✓(*)	✓	✓	✓	+	-	✓(*)
	#2.2;1.2 accept/reject suggestion	✓(*)	✓	✓	✓	✓	-	✓(*)
	#2.5 add metadata to mapping	-	✓	✓	✓	-	-	-
	#2.3 move a mapping to list	-	✓	✓	✓	+	-	-
	#5.0 ontology	✓	-	✓	✓	-	-	-
inspect	#2.2;1.4 mapping suggestions	✓(*)	✓	✓	✓	+	-	✓(*)
	#2.3 mappings	✓(*)	✓	✓	✓	✓	✓	✓(*)
	#4.4 heavy-regions	✓	-	-	✓	-	-	+
	#2.4 filter/search	-/✓	-/✓	-/-	✓/✓	-/-	+/✓	-/-
	#4.1/2/3;2.1;1.1/3 ontologies	✓	✓	✓	✓	✓	+	✓
explain	#4.2/7;5.8 why/how suggested	+	+	✓	✓	+	+	+
	#4.3 why accepted	-	✓	✓	✓	-	-	-
	#4.5 starting point	+	-	-	+	✓	-	+
	#4.6 process state	✓	+	+	✓	+	-	+
large-scale	#5.1;3.3 sessions	+	✓	+	+	+	-	+
	#5.2 clustering	✓	+	-	✓	✓	✓	✓
	#5.3 reduce user interventions	-	+	+	-	-	-	-
	#5.4 collaboration	-	-	-	-	-	-	-
	#5.5 environment	-	+	+	-	-	+	+
	#5.6 recommend/rank	-	✓	+	+	✓	-	✓
	#5.7;3.4 debugging	-	✓	✓	✓	✓	✓	-
	#5.8;4.2/7 matchers configuration	-	✓	-	-	✓	✓	✓
	#5.9.1;3.2 trial execution	-	-	-	-	-	-	-
	#5.9.2;1.1 temporary decisions	✓	+	+	✓	-	-	-

Table 2. Requirements to support user involvement in large-scale matching tasks. (supported(✓); partly supported(+); special case, details in the text(*); not supported(-))

the representation in the other. During the alignment process each ontology is represented as a pair of views—a tree and a small world graph—i.e., four in total. The trees provide well-known editing and exploratory functionalities. There is no clear distinction between mappings and mapping suggestions (✓(*)). Mappings are edited, accepted and rejected in the tree views by toolbar buttons for defining the type of mappings. The small world graphs represent an ontology as a graph where the nodes (represent the entities) are clustered according to a selected level of detail. The size of the clusters corresponds to the number of nodes in them. The edges between the clusters represent the selected relation (mutual property). Intuitive exploration is achieved by the linking and brushing technique, adjustable level of details (by means of a slider) and selecting a relationship to present (from a drop-down list). The small world graphs provide and overview of the ontologies where color-coding provides an overview of the similar clusters (in the two ontologies) and the colors of the clusters are inherited from the underlying nodes according to one (out of three) strategy. Tooltips and labels can be switched on and off.

Different sessions are not directly supported, but simple interruption and resumption of the alignment process can be achieved by saving and loading the input file which

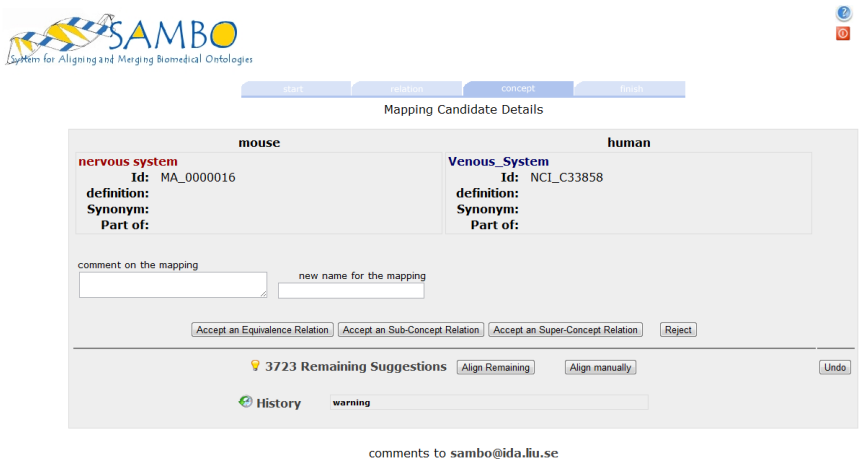


Fig. 1. SAMBO [13].

contains the mappings. Temporary decisions for questionable mappings are supported by a tracking button. Undo/redo buttons and history of activities are also provided.

3.2 SAMBO

SAMBO [13] (based on [14]) is an ontology alignment system that addresses the challenges related to user involvement by introducing interruptible sessions—computation, validation and recommendation sessions. The computation session computes mapping suggestions between two ontologies and can utilize results from previous validation and recommendation sessions. The user validates the mapping suggestions during the validation session. A reasoner may be used during both sessions to check the consistency of the (validated) mapping suggestions in connection with the ontologies. Both sessions can provide partial results upon interruption thus the validation session may start before the end of the computation and not all of the mapping suggestions need to be validated at once. The recommendation session matches small parts of the two ontologies offline using an oracle or previous validation decisions if available and employs different (combination of) algorithms and filtering strategies in order to recommend the best future settings for matching the two ontologies. The results of the sessions are stored in a database. The user may choose to start a new or to resume a saved session.

The user interface allows selection of matchers, their weights and strategies for combination. Two alternating modes are available during the validation—suggestion (shown in Figure 1) and manual mode. All suggestions for a concept are shown at once during the suggestion mode. The user can give a name for and annotate a mapping/concept. The user can accept/reject a suggestion by pressing a dedicated button. Both ontologies are shown as indented trees during the manual mode and the user can create a mapping by selecting a concept in each tree. A search function is implemented for locating a term of interest. Lists with the previous accepted/rejected and remaining suggestions are available. An undo button is available as well.

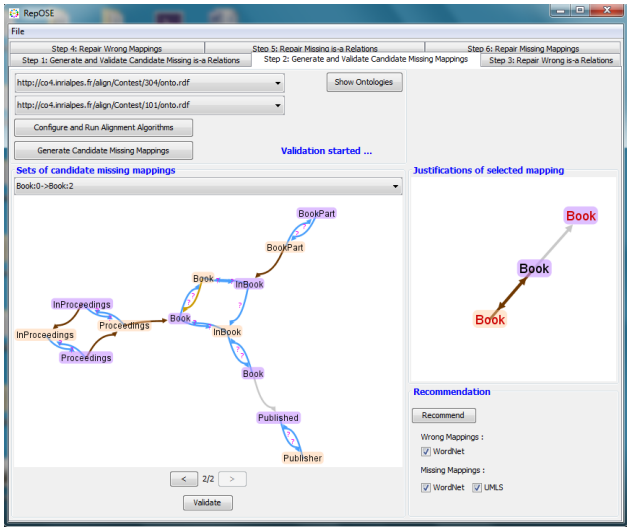


Fig. 2. RepOSE [12].

3.3 RepOSE

RepOSE [12], shown in Figure 2, is based on an integrated taxonomy alignment and debugging framework. The system can be seen as an ontology alignment system with a debugging component for detecting and repairing modelling defects in taxonomy networks (missing and wrong subsumption relations/mappings). The alignment process goes through three phases—generation of mapping suggestion, validation and repairing. Separate panels are provided for the validation and repairing phases to guide the user through them. Possible starting points, recommendations and ranking strategies are available during both phases. The alignment process can be configured by selecting matchers, their weights and the threshold for filtering the mapping suggestions. The algorithm for detecting defects in the debugging component can be seen as a structure-based alignment algorithm—as such it is configured separately. The suggestions computed from it are logically derivable and they are presented to the user together with their derivation paths. The rest are only presented with their confidence values.

During the validation phase the mapping suggestions are shown as graphs in groups where the last group in the list contains the most suggestions. The nodes in the graph represent concepts and the edges—relations and mappings. The nodes are color-coded according to their hosting ontology and the edges—the state of the represented relations/ mappings—mapping suggestions, asserted/added/removed relations/mappings. When the user accepts/rejects a suggestion the corresponding edge is labeled accordingly and it is moved to the list for repairing. The user can validate only a portion of the suggestions and start the repairing phase. The user can see each pair of ontologies and their current alignment and the entire ontology network upon request. During the repairing phase the system provides alternative repairing actions instead of directly adding the validated mapping. Logically derivable wrong mappings can be also repaired.

The system checks for contradictions after each group of suggestions is validated and after a repairing action and warns the user if such are found. There is no indication for the process state but it can be observed by reflecting on the validation and repairing phases. Sessions are only supported through saving/loading the ontologies and mappings, but the suggestions are not preserved and have to be computed from scratch.

3.4 AML

AML has been designed based on AgreementMaker with the purpose of matching very large ontologies. Its user interface is presented in [21]. The working area in AML is divided into two panels—a Resource Panel, on the top, provides a summary of the ontologies, alignment, etc., and a Mapping Viewer where modules extracted from the ontologies and alignment are represented as graphs. Instead of showing the entire network, the visualization is focused on a single mapping where the graph depicts the mapping, up to five (default is two) levels of ascending/descending concepts of the concepts in the mapping and other mappings between the displayed concepts (if any). The nodes and edges are labeled with the names of the classes and relations (subsumptions are not labeled), respectively, and colored depending on the ontology they belong to. The mappings are labeled with their confidence values and their directions are denoted with arrows. Three options are provided for navigating through the mappings—list of mappings, previous/next buttons and search (in combination with auto-complete). The user can configure the alignment process by selecting a matcher, its threshold, cardinality for the alignment and sources of background knowledge. The final alignment can be repaired and evaluated against a reference alignment.

3.5 COMA++

COMA++ is an alignment system for matching large schemata and ontologies [1]. The system consists of five components accessible through a user interface [4]. The *repository* stores the ontologies and alignments. The Workspace tab provides access to the *schema and mapping pools* which manage the ontologies and alignments in memory. Other operations involving alignments, such as merging schema and alignments, add/remove mappings in edit mode, comparing (evaluating an alignment against a reference alignment using different quality measures) and diff/intersect (determining the different/shared mappings between two alignments) are provided as well. The Match menu provides a variety of options for configuring the matching process through the *match customizer*—creating/modifying/deleting/resetting matchers and strategies, showing the dependencies between them and saving them (into the repository) for future use. The matching process is performed in the *execution engine*. Some of the strategies support iterations, where the user can modify the output prior to the execution of the next iteration. The toolbar has buttons for configuring/running/interrupting the process, step-by-step execution and editing mappings. The ontologies are shown side-by-side as unmodifiable indented trees and the mappings between them are represented as lines color-coded depending on their confidence values. There is no clear distinction between mappings and mapping suggestions (✓(*)). The highest confidence value is assigned to the manually created mappings. The regions with many mappings can be observed by

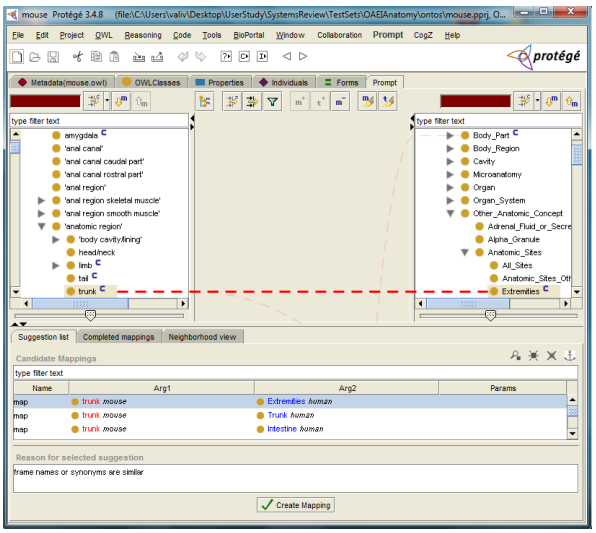


Fig. 3. The latest version of CogZ.

the high number of lines between them. The process state can be observed by the predominant color of the mappings. Sessions are supported as in RepOSE.

3.6 PROMPT

The PROMPT suite [18] is a set of Protégé plug-ins for managing ontologies and their versions: iPROMPT merges and aligns ontologies interactively employing the local context of the concepts; AnchorPROMPT computes additional mapping suggestions acting on a larger scale than iPROMPT; PROMPTDiff performs structural comparison between different versions of an ontology and PROMPTFactor extracts an independent modules from an ontology. These plug-ins share interface components, data structures, some algorithms and heuristics.

The first version of PROMPT, [17], shows the source and target ontologies as indented trees on both sides of the screen where the mapping suggestions are presented as a list of pairs between them. An explanation for why this pair is a mapping suggestion is provided to the user. The user can examine the suggestions from the list, save those that are correct or create new mappings. Upon user action the tool detects conflicts, if any it suggests solutions and generates new suggestions in the area the latest operation has happened. The suggestions/conflicts are resorted to first list those in the area of the latest operation. PROMPT can log operations and execute them again if needed. The process state can be observed indirectly.

3.7 CogZ

CogZ, shown in Figure 3, addresses the cognitive support requirements from [6]. It is a visualization plug-in which extends the PROMPT user interface and reuses the rest of its components.

The first version of CogZ, Jambaprompt, includes a graph visualization of the neighborhood of each of the concepts in a mapping suggestion—direct super and subclasses. Each of the classes can be expanded thus providing an incremental navigation. The Jambaprompt plug-in also supports filtering of the mapping suggestions by various criteria. It is extended in [6] to provide an overview of the ontologies and mappings by employing treemaps. The user can identify potentially 'heavy regions' using the treemaps in combination with color-coding. Pie-charts provide additional details regarding already mapped concepts and mapping suggestions. Temporary mappings, different from the mapping suggestions, are introduced in CogZ to relieve the users' memory and help them to write down potential solutions. Similarly to COMA++, the mappings between the ontologies (shown as trees) are presented with lines which can be annotated to provide additional details. CogZ provides semantic zoom and interactive search.

4 Discussion

Table 2 shows the systems' support for the requirements identified in section 2. The manipulation and inspection requirements are almost entirely supported by the first four systems. However to be able to draw conclusions for the level of usability of the different visualization approaches, a user study is needed. It is worth noting that COMA++ and AlViz do not distinguish between mappings and mapping suggestions, a functionality that may help the users to keep track which correspondences have been already visited. The least supported category from the requirements in [6] is the one that assists the users most in understanding the reasons for suggesting/accepting mapping suggestions—while PROMPT and CogZ provide a textual description to explain the origin of mapping suggestions, the other tools only present a confidence value (which may (not) be enough depending on how familiar the domain expert already is with the ontology alignment field). Other requirements in this category include providing a starting point and a state of the process. Even though rarely supported they can often be observed by the number/status of the verified suggestions.

Some systems limit the amount of data presented to the user by using sessions and clustering. Only one system preserves the state of the process during interruptions. The others partially address the session requirement by save/load (ontologies and alignments) functions but without preserving the already computed suggestions. Almost all of the tools support clustering of the content presented to the user (not necessary for all views/modes) to avoid cluttering of the display, clustering during the computations is also often supported. Another possibility could be to guide the user (through complex interfaces and huge input) by presenting different interfaces connected to different phases of the process, for instance, by providing a different view for each phase. The existence of different phases in general could also allow for more opportunities for fine-tuning of the process.

The session-based approach in [13] helps reducing the user interventions during the alignment process by reusing previously validated mappings. PROMPT takes into account the area of the latest user intervention while computing a new portion of suggestions to maintain the user's focus. To assist the user decision making process some systems provide recommendations in various forms—SAMBO provides a recommendation session, COMA++ default matchers configuration, RepOSE recommendations (from external sources) during the validation. Matchers' configuration is also supported to different extent—COMA++ provides advanced matchers' combinations while RepOSE only supplies a list with matchers and their weights. To support temporary decisions CogZ introduces temporary mappings and AlViz a tracking button. SAMBO partially presents such functionality by an undo button and history of actions, PROMPT by reapplying the user actions. Trial execution is not supported by any of the tools.

Looking at the table we can conclude that most of the systems provide debugging techniques, but this is not the case in reality as discussed in subsection 2.2. Although these systems consider debugging of the alignment, they address different kinds of defects—RepOSE detects/repairs modelling defects in taxonomies, SAMBO checks for inconsistencies and AML addresses disjointness assuming the ontologies are coherent. Further, RepOSE relies on manual repairing while AML repairs the alignment automatically.

The social and collaborative matching is still a challenge. SAMBO, PROMPT and CogZ provide mapping annotations but it is unlikely they have been developed to address this issue. While implementing other functionalities SAMBO and COMA++ took first steps in providing a collaborative environment by introducing permanent storages. AML, PROMPT and COMA++ have functions for evaluating an alignment against a reference alignment and for comparing two alignments.

5 Conclusions

This paper defines a set of requirements to address the user involvement in large-scale ontology alignment tasks. It provides a literature based overview of several systems selected due to their mature interfaces and features that address the alignment of large ontologies.

Since the papers describing the systems mostly focus on algorithms and rarely on user interfaces such assessment of the coverage of the requirements is inherently imprecise. In order to provide better understanding for how the systems support the requirements identified in section 2 we intend to conduct an observational user study as a future work. The study will consider the requirements in the manipulation, inspection and explanation categories by developing tasks that address them in a large-scale setting. It will provide detailed overview of the advantages and disadvantages of the user interfaces of several selected systems. Changes in the list with requirements may occur as a consequence of the study.

Acknowledgments. We thank the National Graduate School in Computer Science (CUGS) and the Swedish e-Science Research Centre (SeRC) for financial support.

References

1. D Aumüller, H H Do, S Maßmann, and E Rahm. Schema and ontology matching with COMA++. In *SIGMOD*, pages 906–908, 2005.
2. I F Cruz, C Stroe, and M Palmonari. Interactive user feedback in ontology matching using signature vectors. In *ICDE*, pages 1321–1324, 2012.
3. B Cuenca Grau et al. Results of the ontology alignment evaluation initiative 2013. In *OM*, pages 61–100, 2013.
4. H H Do. *Schema Matching and Mapping-based Data Integration*. PhD thesis, 2005.
5. J Euzenat and P Shvaiko. User Involvement. In *Ontology Matching*, pages 353–375. 2013.
6. S M Falconer and M D Storey. A Cognitive Support Framework for Ontology Mapping. In *ISWC/ASWC*, pages 114–127, 2007.
7. M Granitzer, V Sabol, K W Onn, et al. Ontology Alignment—A Survey with Focus on Visually Supported Semi-Automatic Techniques. *Future Internet*, pages 238–258, 2010.
8. V Ivanova, J L Bergman, U Hammerling, and P Lambrix. Debugging taxonomies and their alignments: the ToxOntology-MeSH use case. In *WoDOOM*, pages 25–36, 2012.
9. E Jiménez-Ruiz, B C Grau, Y Zhou, and I Horrocks. Large-scale Interactive Ontology Matching: Algorithms and Implementation. In *ECAI*, pages 444–449, 2012.
10. E Jiménez-Ruiz, C Meilicke, B C Grau, and I Horrocks. Evaluating Mapping Repair Systems with Large Biomedical Ontologies. In *Description Logics*, pages 246–257, 2013.
11. T Kirsten, A Gross, et al. GOMMA: a component-based infrastructure for managing and analyzing life science ontologies and their evolution. *Journal of Biomedical Semantics*, 2(1), 2011.
12. P Lambrix and V Ivanova. A unified approach for debugging is-a structure and mappings in networked taxonomies. *Journal of Biomedical Semantics*, 4:10, 2013.
13. P Lambrix and R Kaliyaperumal. A Session-Based Approach for Aligning Large Ontologies. In *ESWC*, pages 46–60. 2013.
14. P Lambrix and H Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.
15. M Lanzenberger, J Sampson, and M Rester. Ontology visualization: Tools and techniques for visual representation of semi-structured meta-data. *J.UCS*, 16(7):1036–1054, 2010.
16. M Lanzenberger, J Sampson, M Rester, Y Naudet, and T Latour. Visual ontology alignment for knowledge sharing and reuse. *Journal of Knowledge Management*, 12(6):102–120, 2008.
17. N F Noy and M A Musen. Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI*, pages 450–455, 2000.
18. N F Noy and M A Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *Journal of Human-Computer Studies*, 59(6):983–1024, 2003.
19. L Otero-Cerdeira, F J Rodríguez-Martínez, and A Gómez-Rodríguez. Ontology matching: A literature review. *Expert Systems with Applications*, 2014.
20. H Paulheim, S Hertling, and D Ritze. Towards Evaluating Interactive Ontology Matching Tools. In *ESWC*, pages 31–45, 2013.
21. C Pesquita, D Faria, E Santos, J Neefs, and F M Couto. Towards Visualizing the Alignment of Large Biomedical Ontologies. In *DILS*, pages 104–111, 2014.
22. E Rahm. Towards large-scale schema and ontology matching. In *Schema matching and mapping*, pages 3–27. 2011.
23. F Shi, J Li, J Tang, G Xie, and H Li. Actively learning ontology matching via user interaction. In *ISWC*, pages 585–600. 2009.
24. P Shvaiko and J Euzenat. Ontology Matching: State of the Art and Future Challenges. *Knowledge and Data Engineering*, 25(1):158–176, 2013.

Sensemaking on Wikipedia by Secondary School Students with SynerScope

W.R. van Hage[1,2], F. Núñez Serrano[2,3], T. Ploeger[1], and J.E. Hoeksema[1,2]

[1] SynerScope B.V.
[2] VU University Amsterdam
[3] Universidad Politécnica de Madrid

Abstract. Visual analytics of linked data can be done by secondary school students with minimal preparation. We study the learning curve of students while answering typical Web analytics questions on Wikipedia and DBpedia using SynerScope visual analytics software. We find that after a short tutorial students are able to answer most complex questions in a few minutes, learning by trial and error. Older students are faster on average, but motivation appears to be a stronger factor than age for success. Answering speed doubles within two hours of experience while correctness increases.

1 Introduction

The world will soon face a critical shortage of data scientists, professionals with analytical expertise that can take advantage of (linked) data to answer questions [7]. One strategy to mitigate this problem is to enable non-experts to take over part of the data science tasks. We pose that data science is comprised of many tasks that do not all require expert-level knowledge. In this article we restrict ourselves to a category of data science sensemaking tasks on Web data that is common in data journalism and involves basic analytics operations, search, and Web browsing. We hypothesise that, given the right tools, untrained people can quickly be trained to do such tasks, avoiding a complete data science education.

The goal of this article is to test this hypothesis by doing an experiment to demonstrate the feasibility of having untrained people do prototypical sensemaking tasks given visual analytics tools. Specifically, we look at secondary school students with no analytical experience, and ask them to answer complex questions about Wikipedia content using the SynerScope[4] visual analytics software illustrated in Figure 1. We want to know if users can get to an answer after a minimal amount of training in the tool. We want to know how long it takes them to find an answer and if their time-to-answer decreases as their experience with the tool increases, and what the influence is of their age and corresponding level of education.

The line of reasoning we follow is that the required skills for such sensemaking data science tasks can be rapidly acquired or substituted with appropriate tools. If this is the case and if SynerScope is an appropriate tool for the task, then we should be able to show that unskilled people can accomplish the sensemaking tasks.

[4] `http://www.synerscope.com`

This idea of empowering people by means of augmented reasoning through human-computer interaction is not new [6], but in recent years the development of interactive tools for visual analytics have intensified. Some of these tools are targeted at programmers (e.g., [1, 10, 11]), while other tools target non-programmers (e.g., [13, 12, 14, 9, 2, 8]). For this experiment we need a tool from the latter category that is network centric and allows search and Web browsing. We use SynerScope [13, 5, 4], one of the tools that meets these requirements.

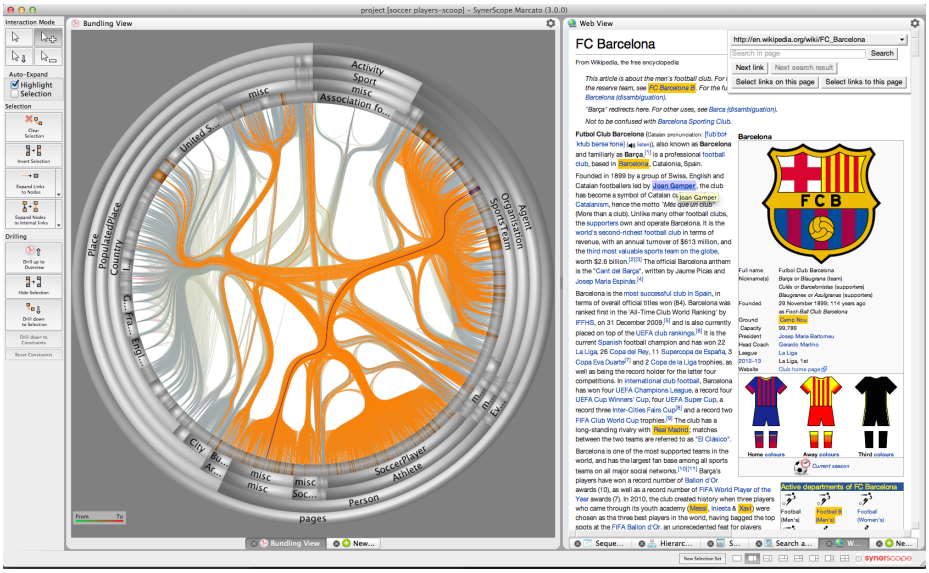


Fig. 1. A screenshot of the SynerScope visual analytics tool showing Wikipedia and DBpedia data. This picture shows two coordinated views: a hierarchical edge bundling network view and a Web browser.

The rest of this paper is organised as follows: Section 2 describes the SynerScope software in more detail. Section 3 outlines the experimental set-up, including the tasks, tooling, and procedure. Section 4 shows our findings. Section 5 discusses our findings, draws conclusions and suggests future work.

2 The SynerScope Software

SynerScope is a visual analytics application that delivers real time interaction with dynamic network-centric data. SynerScope supports simultaneous visualisations and coordinates user interaction, enabling the user to identify causal relationships and to uncover unforeseen connections.

The central interaction paradigm of SynerScope is Multiple and Coordinated Views. SynerScope shows a number of different perspectives on data, for example, relations

and time, and each selection made in either of these views causes an equivalent selection to be made in all other views. This enables the user to explore correlations between different facets of data.

SynerScope is designed to work with a very basic information schema. This schema consists of two object types: Nodes and Links. Links connect two Nodes. Both Nodes and Links can have additional attributes of a number of data types, including integers, floating point numbers, free text, date and time, latitude and longitude.

What follows is a short overview of each visualisation that is offered by SynerScope.

Table View The Table View provides a traditional spreadsheet view on the data. For each type of Node and each type of Link, there is a separate sheet. The Table View shows all the data as a table of values.

Hierarchical Edge Bundling View The Hierarchical Edge Bundling View (HEB) is the primary network view in SynerScope. Each Node is visualised as a point on a circle, and each Link is visualised as a curved line between its source and target Node.

The Nodes are grouped hierarchically, based on one or more of their attributes. The Links between Nodes of the same hierarchical category are bundled together (as if they were tied together with a cable tie).

Massive Sequence View The Massive Sequence View (MSV) is the primary temporal view in SynerScope. Each Node gets a fixed position on the horizontal axis. Nodes are grouped hierarchically in the same fashion as in the HEB. Links between Nodes are represented by a horizontal line between the respective positions of the Nodes. On the vertical axis the user can select a scalar attribute, typically a time or date. This orders the Links temporally.

Map View The Map View is the primary spatial view in SynerScope. The user can select two attributes from any Node or Link data source to interpret as WGS84 latitude and longitude coordinates. These attributes are used to plot the Nodes (not the Links) on a map as points.

Scatter Plot View The Scatter Plot View uses Cartesian coordinates to relate the values of two attributes of either Nodes or Links. Dots are drawn on a two-dimensional chart, the positioning relative to the horizontal and vertical axis being determined by the attribute's values. A third attribute can used to set the size of the dots.

Search and Filter View The Search and Filter View is an interactive view that allows the user to select Nodes or Links by searching by value.

Web View The Web View is an interactive view that allows the user to view any URL's that are an attribute of a node or a link.

The user can interact with SynerScope's views in several ways: By selecting and highlighting data, drilling down to or up from a selection, and expanding selections from nodes to connected links or vice versa. Every interaction method is coordinated across multiple views.

3 Experimental Set-up

Sensemaking Tasks In the experiment we look at 10 exemplar Web analytics questions that each require a combination of at least two of the following operations to answer: network navigation, filtering on categorical and numerical variables, grouping and counting, search, Web browsing within Wikipedia, and zooming in on data selections. Examples of the questions are: *"How many former AFC Ajax soccer players died in Paramaribo and what was the cause of death?"*, or *"Which page about a disease is linked to most from pages about physicists?"*. The complete set of questions can be found on FigShare [15]. Question number 8 is marked as a difficult question, because it is the only question that involves a set intersection between two sets of network patterns.

SynerScope Visual Analytics Tooling The SynerScope tool used by the students is a graphically accelerated visual analytics application that combines a number of views on networked data. It offers real-time interactive exploration using scatter plots, timelines, maps, hierarchical edge bundling network layouts, an integrated Web browser, a search engine, and a spreadsheet table view. The selections made in any of these views are propagated to all the other views. A video illustrating interaction with the Wikipedia data can be found on FigShare [3].

Procedure The experiment consists of five parts: (1) a 30m plenary introduction to the experiment and the data sets used, (2) a 15m plenary tutorial to the SynerScope visual analytics tool, (3 and 4) two 45m sessions where students try to answer questions using SynerScope, (5) a concluding discussion and personal interviews. The students are asked to answer as many as possible of 10 questions about 3 subsets of Wikipedia within 90m. Each set centers around pages on a specific topic.

Data Sets The topics covered in the experiment are: (1) Athletes classified as soccer players and trainers of AFC Ajax, FC Barcelona, and Manchester United, (2) Scientists classified as physicist, (3) Artists in the pop genre. Each of these three sets consist of around 3000 Wikipedia pages about the topic (the seed set), all the pages that are linked to from the seed pages (the "out" context), all pages that link to the seed pages (the "in" context), and all the links between the seed, "out" context, and "in" context pages. This amounts to three sets of around 100k–200k pages and 300k–500k page links. Each page is assigned around 18 attributes with information about the page, such as the page title, the number of words on the page, the in degree and out degree, a three-level hierarchical topic classification of the main subject of the page (e.g. Actor-Artist-Person, or Building-ArchitecturalStructure-Place) derived from the DBpedia rdf:type property of the corresponding DBpedia resource, birth/death date and place, and topic-specific properties such as respectively soccer team, university, or band. An example of the three schemas can be found in the hand-outs for the students [15]. We made a selection of the DBpedia types (downloaded september 2013) that form a hierarchical partitioning of the Wikipedia pages. We only considered types from the DBpedia ontology, ignoring other type hierarchies such as Yago, FreeBase, and Schema.org. The selection process involved dividing the types into three hierarchical layers, and imposing a preferential

ordering onto the types. For example, Amsterdam was assigned City at level 1, PopulatedPlace at level 2, and Place at level 3, discarding types such as Settlement to form a proper partition. When type information is missing, a placeholder type is assigned.

Test Subjects The students involved as test subjects in the experiment are 63 middle school and high school students (9 female, 54 male) from three schools in the Amsterdam area between the ages of 12 and 18, divided into 34 groups of size 1–3. The experiments were performed in two labs of the VU University Amsterdam Network Institute.[5] One running SynerScope in the Amazon cloud accessed through a Web-based client (OTOY), the other running SynerScope natively on gaming PCs with modern NVIDIA GeForce GPUs. The students were paired up and given a hand-out describing the three data sets, listing all the questions, and containing a form to record the answers and the time taken [15]. During the experiment students were assisted by answering specific technical questions, but were given no other guidance that would help them find answers.

4 Results

There was a large variation in the productivity of the various students, as can be seen in Figure 2. This can be expected of students that have no intrinsic motivation to cooperate in the experiment. The motivated students answered all questions, while two groups did nothing and are excluded from the results. In general the total number of 10 questions was too high to answer for most students in two 45 minute sessions. Most students managed to answer the questions of two topics (6 or 7 questions). Of the questions that were answered, about 60% was answered correctly. There was a large variation, depending on the difficulty of the question. This is illustrated in Figure 3. Some questions were answered partially. For example, when asked for a number and explanation only the number or the explanation was answered correctly. We performed significance tests for the differences in duration between all the categories shown in Figure 2 with a Welch's t-test, and similarly for the categories in Figure 4. There was a slight increase in the number of questions that were answered correctly over time. This trend is significant according to a Mann-Kendall test ($p = 0.0318$), even when counting partial answers as false answers. Students performed faster and more consistently for subsequent questions. This is illustrated in Figure 4 (right), specifically with questions 1–7 which were consistently answered before time ran out. This increase in speed is significant between the first and last of the questions in the sequence at a confidence level of 95%. Older students seemed to be faster than younger students, but their answers were of a comparable correctness. Although the difference in mean time taken between the fastest and slowest age groups is a factor 2, a Mann-Kendall test does not show a significant downward trend ($p = 0.178$). This is due to the relatively small number of observations (34 student teams) and a class of particularly talented middle school freshmen that performed on par with 18-year-olds, but with a significantly higher accuracy. The data used to derive these conclusions can be found on Figshare [15].

[5] Network Institute Tech Labs, `http://www.networkinstitute.org/tech-labs/`

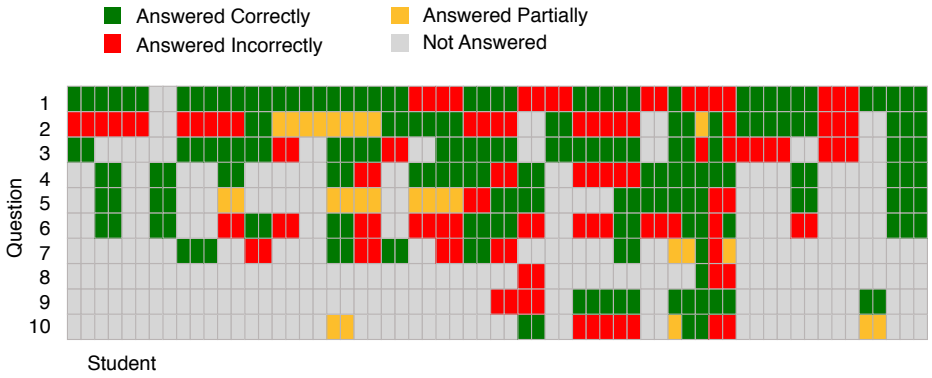


Fig. 2. An overview of the completeness and correctness of the answer to each question by each student. Each column represents the answers given by a student. Each row represents one of the 10 questions. Roughly 55% of the questions were answered, about 60% of the answers were correct.

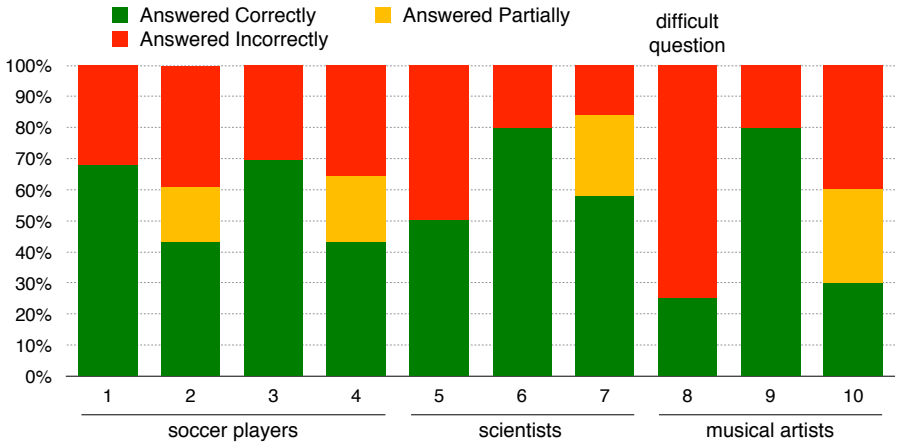


Fig. 3. An aggregation of the correctness of the answers per question. For the questions that were consistently answered (1–7) in the 90m experiment is a rising trend in the quality of the answers. Most students ran out of time before attempting question 8–10.

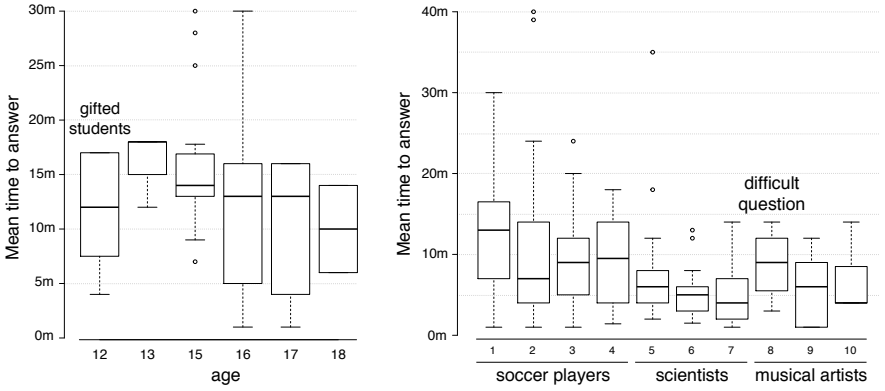


Fig. 4. (left) Time taken to answer a question, aggregated over all questions per age. Older students are faster than younger students with the exception of a group of gifted 12-year-olds; (right) Aggregated time taken to answer each of the 10 questions. There is a non-significant decreasing trend in the time taken per question.

5 Discussion

Given the preliminary nature of these results, we can not draw very strong conclusions yet. If we had more test subjects, we could have repeated the experiment with the topics offered in a randomized order, which would strengthen the conclusions by removing the learning effect and topic preference between the various topics.

We are impressed by what our young test subjects were able to achieve. When given the right tools, visual analytics of linked data really can be done by secondary school students with minimal preparation. We found that after a short tutorial students are able to answer most complex questions in a few minutes, learning by trial and error. Within two hours of experience, answering speed doubles within while correctness increases.

The older test subjects more frequently asked for help when they get stuck than the younger test subjects, who just found their own way through trial and error, and therefore also take longer to get to an anser than the older students (as can be seen in Section 4. Overall, motivation appears to be a stronger success factor than age. This belief is hard to make concrete, but it is reinforced by our observation that students are quick to accept their first findings as a definitive answer to the question they were working on. When students found information they thought was the right answer, they were fairly quick to accept that answer and wanted to move on to the next question as soon as possible. In contrast to professionals, the students did not verify their answers. For instance, when the students had to find out how many AFX Ajax soccer players died in Paramaribo, they typically accepted all the soccer players that died in Paramaribo as an answer, without checking if they played in AFC Ajax. We think this can be explained by the lack of feedback during the experiment. Students were not penalised for wrong answers or rewarded for right answers, and the experiment was a one time encounter with the software. We expect that many of the incorrect or partial answers could have been improved if the students were to have verified their answers.

The experiment reinforced our belief that visual analytics software must be highly interactive and present immediate feedback to the user. During the interviews at the end of the session students were generally positive about the software and tasks and thought the experiment gave them a new perspective on Wikipedia. Their main negative remark was that SynerScope running on Amazon was distractingly slow. In actuality, the software was equally fast on Amazon instances as on local machines, but the lag introduced by network congestion, network latency, and video compression, removed the sensation of true interactivity. In isolated cases, for example, when zooming out to the entire data set of 400k links, students had to wait a few seconds. Delays in interaction like these appeared to interrupt the student's train of thought.

We found that students of all ages are able to effectively use the SynerScope tool to answer the questions. Older students are usually faster, but not significantly more accurate. We would like to further test these findings with older and younger subjects.

Acknowledgements

Thanks go to the Damstede, Pieter Nieuwland College, and Cygnus Gymnasium schools for their participation in this experiment. We thank the VU Network Institute for the use of their facilities, and Samir Naaimi for his assistance during the experiments. This work was done within the context of the SAGAN project supported by ONR Global NICOP grant N62909-14-1-N030, the EU FP7 NewsReader project (316404), and the Dutch COMMIT Data2Semantics project.

References

1. D3.js: D3.js - data-driven documents (2014), http://d3js.org/
2. Gapminder: Gapminder: Unveiling the beauty of statistics for a fact based world view (2014), http://www.gapminder.org/
3. van Hage, W.R.: SynerScope on Wikipedia (movie) (06 2014), http://dx.doi.org/10.6084/m9.figshare.1061499
4. Holten, D., Cornelissen, B., van Wijk, J.: Trace Visualization Using Hierarchical Edge Bundles and Massive Sequence Views. In: Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on. pp. 47–54 (June 2007)
5. Holten, D.: Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. IEEE Transactions on Visualization and Computer Graphics 12(5), 741–748 (Sep 2006), http://dx.doi.org/10.1109/TVCG.2006.147
6. Licklider, J.C.R.: Man-computer symbiosis. Human Factors in Electronics, IRE Transactions on (1), 4–11 (1960)
7. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A.: Big data: The next frontier for innovation, competition, and productivity (2011), http://www.mckinsey.com/insights/business_technology/big_data_the_next_frontier_for_innovation
8. Paulheim, H.: Explain-a-lod: Using linked open data for interpreting statistics. In: Proceedings of the 2012 ACM international conference on Intelligent User Interfaces. pp. 313–314. ACM (2012), http://www.ke.tu-darmstadt.de/resources/explain-a-lod

9. Qlikview: Business Intelligence and Data Visualization Software — Qlik (2014), http://www.qlik.com/
10. R: The R Project for Statistical Computing (2014), http://www.r-project.org/
11. Skjæveland, M.G.: Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In: Extended Semantic Web Conference (2012), http://dev.data2000.no/sgvizler/
12. Spotfire, T.: TIBCO Spotfire - Business Intelligence Analytics Software & Data Visualization (2014), http://spotfire.tibco.com/
13. SynerScope: SynerScope — Connecting the dots (2014), http://www.synerscope.com/
14. Tableau: Business Intelligence and Analytics — Tableau Software (2014), http://www.tableausoftware.com/
15. van Hage, W.R., Ploeger, T., Hoeksema, J., Núñez Serrano, F.: Wikipedia SynerScope experiment (06 2014), http://dx.doi.org/10.6084/m9.figshare.1060254

Towards a Visual Annotation Tool for End-User Semantic Content Authoring

Torgeir Lebesbye and Ahmet Soylu

Department of Informatics, University of Oslo, Norway
{torgeirl, ahmets}@ifi.uio.no

Abstract. There is a great amount of data on the Web, but to a large extent it is not published as linked data that computers can consume. Visual annotation tools have a considerable potential to empower end users to contribute their data to the Semantic Web, and could prove to be a solution to get more data on the Web linked. To this end, numerous tools have been developed; however, there still remains challenges to be addressed. In this paper, we present and discuss a set of prominent requirements toward the realisation of a visual annotation tool for end-user semantic content authoring.

Keywords: Semantic Content Authoring, Direct Manipulation Interface, End-User Development

1 Introduction

When Berners-Lee invented the Web in 1989, his motivation was to allow people to share and link documents without the barriers of hardware, file systems or data formats [4]. It later evolved into the Social Web, referred to as Web 2.0, where anyone could be the producers of contents through blogs, wikis and social media, and it became easy for people to collaborate on the Web.

In later years, the introduction of semantic technologies has made it possible to describe the meaning of data in a language more consumable for computers: *words* written with markup languages like XML[1], *grammar structure* using RDF[2], and *logic* described in knowledge languages like OWL[3] [7]. This means a contextual Web where data in documents are linked and can be mashed with data from other sources in a completely new way. It is called the Semantic Web, often referred to as Web 3.0.

So-called lowercase semantic technologies enabled linked data to be added to documents with in-content annotations [1]. Microformats[4] is a widely used family of data formats that includes hCard, hCalendar and hAtom. eRDF was W3C's original attempt at simplifying annotation, but they abandoned it for

[1] http://www.w3.org/TR/xml/
[2] http://www.w3.org/TR/rdf-schema/
[3] http://www.w3.org/TR/owl-primer/
[4] microformats.org

RDFa[5]. Microdata[6] is an alternative to RDFa originating from the WHATWG initiative[7].

While linking documents and creating social content is easy and accessible through numerous tools and services, adding semantics to a web document requires knowledge of the involved technologies and a large technical skill-set that most end users often do not possess. Most of the data published on the Web isn't linked, and much of this data is managed by end users within organisations and on the open Web. Therefore, if we are to convert the current Web dominated by unstructured documents into a Web of Data, end-user involvement has a crucial role to play. In this respect, visual annotation tools have a considerable potential to empower end users to contribute their data to the Semantic Web, and could prove to be a solution to get more data on the Web linked.

To this end, numerous tools have been developed [11]; however, there still remain challenges to be addressed. In this paper, we present and discuss a set of prominent requirements towards the realisation of a visual annotation tool for end-user semantic content authoring. For each requirement, we shortly discuss how it improves a visual annotation tool from an end-user perspective. We believe that the discussion presented in this paper may be useful for researchers and practitioners working on annotation tools for semantic content authoring.

The rest of the paper is organised as follows: Section 2 discusses design requirements for a visual annotation tool for end users. Section 3 looks at related work, and finally Section 4 concludes the paper.

2 Design Requirements

End-User Development (EUD) allows users to act as non-professional software developers, creating, modifying, or extending software artefacts [14]. It includes spreadsheets and filters for emails, and is something more and more users do without thinking of it as software development. Not surprisingly, studies has shown the number of end-user programmers vastly outnumber professional programmers [19] and estimate it will continue to do so in the future. Annotation tools for end users are meant to enable end users to modify and extend software artefacts, and could be considered within EUD.

A visual annotation tool employs a direct manipulation approach [20], where end users can directly manipulate visual objects representing domain elements and application functionality to incorporate semantic knowledge, rather than dealing with a command language. We can assume that users managing web pages have at least some domain knowledge. An annotation tool should empower them to access and use this domain knowledge without requiring expertise in web technologies and ontologies, and should take the following into account:

[5] http://www.w3.org/TR/rdfa-syntax/
[6] http://www.w3.org/TR/microdata/
[7] https://whatwg.org/

2.1 Bottom-up

Semantic content authoring tools are often divided into two main categories: top-down and bottom-up [11]. A top-down approach focuses on making and extending ontologies during the annotation process, while a bottom-up approach focuses on the document and uses existing ontologies to annotate the document.

End users are expected to have little or no experience with ontologies, therefore a bottom-up approach is preferable. Moreover, lifting unstructured content to a semantic level is an important issue, given that today the Web is dominated by unstructured documents.

2.2 Human-driven

Another issue is the level of automation – machine-driven vs. human-driven approaches. Some tools detect and annotate text automatically largely based on natural language processing (NLP) techniques [13]. Others provide suggestions, and keep the document valid during the editing process. Machine-driven approaches have a huge advantage in annotation speed, while human-driven approaches hold a higher annotation quality. End users might have little experience with semantic technologies, but as managers of the content they more often have some domain-knowledge. A full automation does not take advantage of this knowledge.

Automation support is particularly important when a large number of documents are involved, yet this should be adequately intertwined with a manual approach. Considering typical users and documents on the Web, a human-driven approach remains more accessible as a generic solution, since automated approaches usually require domain-specific configurations.

2.3 Exploration support

An annotation tool for end users should feature an exploration support, where the underlying ontologies and data can be explored visually. In this respect, visualisations are a powerful way to make the content of a service or tool accessible. For end users, visualisations can make it easier to understand the underlying domain in terms of concepts, properties, and instances.

The big challenge of ontology visualisation is scalability. Ontologies vary in size, from a few hundred nodes to hundreds of thousands of nodes. The vast size of a large ontology can be intimidating, and very hard to get an overview of. A survey on ontology visualisation methods [10] suggests that visualisations should be coupled with effective search functionality and take advantage of semantic data and user data to make the ontology exploration more efficient.

2.4 Complete editing suite

Some annotation tools available today remain primitive, lack important functionality and expressivity. Many of the fully automated tools do not preserve

change in a document. Some tools do not support different in-content annotation technologies and most do not support linking entities through objects' properties. Having said that, a visual annotation tool is not expected to be fully expressive, as certain functionalities and ontology constructs are difficult to grasp even in a visual form.

An annotation tool should preserve change, support the most important formats, and allow for full editing of the document data, while adequately managing the trade-off between usability and expressivity. To this end, often and commonly used functionality and ontology constructs have to be identified and classified with respect to their complexity, as perceived by the end users.

2.5 Usability evaluation

Most of the tools developed in academia undergo little end-usability evaluation, and user studies is often limited to students affiliated with the research groups. Qualitative end-user studies measure whether a tool is competent of meeting its identified aim with respect to a set of criteria, such as effectiveness (i.e., completeness and accuracy) and efficiency (i.e., the cost associated such as time), user satisfaction, learnability etc. [5].

A successful validation of usability and functionality requires user studies to be conducted with users that match the profile of target end users. However, one also needs to be aware that a single summative study only at the end is not sufficient; several formative end-user studies should be held with intermediary prototypes during the design and development process for timely identification of any usability problems.

3 Related Work

Existing tools vary in approach, level of automation, and domain-dependency, in what follows we address only the most prominent ones. Most tools extend the software that the targeted users already uses, to piggyback on concepts and workflow that the users already are familiar with, such as widely used WYSIWYG editor TinyMCE[8] and wiki tool MediaWiki[9].

HayStack semantic blogging [9] and semiBlog [16] were early works on enabling end users to add metadata about the structure and content of their blog posts, their relations (in reply), and subscription to blogs using RSS. DataPress [3] and LinkedBlog [18] are both extensions of the WordPress blogging tool, and the annotation tools are integrated as WYSIWYG editors.

SweetWiki [6] and Semantic MediaWiki [8] are semantic content authoring tools built as wikis that follow the bottom-up approach. They are both based on MediaWiki, a wiki tool originating from the Wikipedia project. They are made specifically for the wiki domain, both with the goal of enhancing the content in the wiki with semantics, while improving search results and other core functionalities.

[8] www.tinymce.com/

[9] https://www.mediawiki.org/

Loomp [15] is an annotation tool for journalists, where the user annotates information fragments, makes mash-ups of those fragments, and keeps track of and reuses them for semantic linking and search.

Annotation tools for Content Management System (CMS) tend to be fully automated. Epiphany [2] uses a web service that finds instances in a web page, and automatically returns a version of the web page with RDFa annotations. It uses the light-box effect often used for image galleries to visualise embedded RDFa. FLERSA [17] is an automated annotation tool built upon Joomla, a popular CMS for building web portals.

OntosFeeder [13] is a web service made to be integrated with a CMS. It supports TinyMCE and FCKeditor on both WordPress and Drupal, is independent of both the editor and the CMS, and is fully automated.

RDFaCE[12] is another extension of the TinyMCE. It has four synchronised views (WYSIWYG editing view, annotations view, fact view, and HTML/RDFa source view), and allows users to switch freely between them during the editing process. It does, however, not include an exploration support of the underlying ontology and does not support linking entities through object properties.

4 Conclusion

In this paper, we have looked at design requirements for a visual annotation tool for end-user semantic content authoring. We believe that such a tool should follow a bottom-up approach, be human-driven, use visualisations to facilitate document annotation and the exploration of the underlying ontologies, and undergo user studies with a representative set of end users. The related work suggests that existing approaches mostly fail to meet these requirements.

Our future work involves design and development of a visual annotation tool for end-user semantic content authoring with these principles in mind.

References

1. B. Adida. hGRDDL: Bridging microformats and RDFa. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):54–60, 2008.
2. B. Adrian, J. Hees, I. Herman, M. Sintek, and A. Dengel. Epiphany: Adaptable rdfa generation linking the web of documents to the web of data. In *Proceedings of the 17th International Conference on Knowledge Engineering: Practice and Patterns (EKAW 2010)*, volume 6317 of *LNAI*, pages 178–192. Springer, 2010.
3. E. Benson, A. Marcus, F. Howahl, and D. Karger. Talking about data: Sharing richly structured information through blogs and wikis. In *Proceedings of the 9th International Semantic Web Conference (ISWC 2010)*, volume 6496 of *LNCS*, pages 48–63. Springer, 2010.
4. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web - a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, 2001.
5. N. Bevan and M. Macleod. Usability measurement in context. *Behaviour and Information Technology*, 13(1-2):132–145, 1994.

6. M. Buffa, F. Gandon, G. Ereteo, P. Sander, and C. Faron. SweetWiki: A semantic wiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):84–97, 2008.
7. B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The Next Step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.
8. D. M. Herzig and B. Ell. Semantic mediawiki in operation: Experiences with building a semantic portal. In *Proceedings of the 9th International Semantic Web Conference (ISWC 2010)*, volume 6497 of *LNCS*, pages 114–128. Springer, 2010.
9. D. R. Karger and D. Quan. What would it mean to blog on the semantic web? In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, volume 3298 of *LNCS*, pages 214–228. Springer, 2004.
10. A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods–a survey. *ACM Computing Surveys*, 39(4):10, 2007.
11. A. Khalili and S. Auer. User interfaces for semantic authoring of textual content: A systematic literature review. *Web Semantics: Science, Services and Agents on the World Wide Web*, 22:1–18, 2013.
12. A. Khalili, S. Auer, and D. Hladky. The RDFa Content Editor - From WYSIWYG to WYSIWYM. In *Proceedings of the IEEE 36th Annual Computer Software and Applications Conference (COMPSAC 2012)*, pages 531–540. IEEE, 2012.
13. A. Klebeck, S. Hellmann, C. Ehrlich, and S. Auer. Ontosfeeder–a versatile semantic context provider for web content authoring. In *Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011)*, volume 6644 of *LNCS*, pages 456–460. Springer, 2011.
14. H. Lieberman, F. Paternó, M. Klann, and V. Wulf. End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paternó, and V. Wulf, editors, *End-User Development*, volume 9 of *Human-Computer Interaction Series*, pages 1–8. Springer, Netherlands, 2006.
15. M. Luczak-Rösch and R. Heese. Linked Data Authoring for Non-Experts. In *Proceedings of the WWW2009 Workshop on Linked Data on the Web (LDOW 2009)*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
16. K. Möller, U. Bojārs, and J. G. Breslin. Using semantics to enhance the blogging experience. In *Proceedings of the 3rd European Semantic Web Conference on the Semantic Web: Research and Applications (ESWC 2006)*, volume 4011 of *LNCS*, pages 679–696. Springer, 2006.
17. J. L. Navarro-Galindo and J. Samos. The FLERSA tool: adding semantics to a web content management system. *International Journal of Web Information Systems*, 8(1):73–126, 2012.
18. I. Ruiz-Rube, C. M. Cornejo, J. M. Dodero, and V. M. García. Development issues on linked data weblog enrichment. In *Proceedings of the 4th International Conference on Metadata and Semantic Research (MTSR 2010)*, volume 108 of *CCIS*, pages 235–246. Springer, 2010.
19. C. Scaffidi, M. Shaw, and B. Myers. Estimating the numbers of end users and end user programmers. In *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2005)*, pages 207–214. IEEE, 2005.
20. B. Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8):57–69, 1983.