Touch-Based Ontology Browsing on Tablets and Surfaces

Touch-based Ontology Browsing on Tablets and Surfaces

Fulvio Corno
Dip. di Automatica e Informatica
Politecnico di Torino
Torino, Italy
fulvio.corno@polito.it

Luigi De Russis
Dip. di Automatica e Informatica
Politecnico di Torino
Torino, Italy
luigi.derussis@polito.it

Luisa Barrera-León
Dip. di Automatica e Informatica
Politecnico di Torino
Torino, Italy
luisa.barrera@polito.it

***Abstract*—Semantic technologies and Linked Data are increasingly adopted as core application modules, in many knowledge domains and involving various stakeholders: ontology engineers, software architects, doctors, employees, etc. Such a diffusion calls for better access to models and data, which should be direct, mobile, visual and time effective. While a relevant core of research efforts investigated the problem of ontology visualization, discovering different paradigms, layouts, and interaction modalities, a few approaches target mobile devices such as tablets and smartphones. Touch interaction, indeed, has the potential of dramatically improving usability of Linked Data and of semantic-based solutions in real-world applications and mash-ups, by enabling direct and tactile interactions with involved knowledge objects. In this paper, we move a step towards touch-based, mobile interfaces for semantic models by presenting an ontology browsing platform for Android devices. We exploit state of the art touch-based interaction paradigms, e.g., pie menus, pinch-to-zoom, etc., to empower effective ontology browsing. Our research mainly focuses on interactions, yet providing support to different visualization approaches thanks to a clear decoupling between model-level operation and visual representations. Presented results include the design and implementation of a working prototype application, as well as a first validation involving habitual users of semantic technologies. Results show a low learning curve and positive reactions to the proposed paradigms, which are perceived as both innovative and useful.**

***Index Terms*—Touch-based interaction, Ontology, Visualization, Browsing, Semantic Web Tool, Mobile**

I. Introduction

Semantic Web technologies and Linked Data are increasingly lying at the core of complex, knowledge intensive applications, involving more and more stakeholders that need to visualize, understand, browse and query information encoded into ontological models, even without deep background in description logic and Web Ontology Language (OWL) formalisms. While issues related to ontology browsing and visualization have been widely investigated by the research community (see [1] for a systematic overview), mostly in the context of ontology editing tools, evidence gathered through observational studies [2] still highlights lack of user satisfaction, in particular for context-level visualization, selective visualization of ontology parts, summaries and overviews. The scenario is further complicated by the variety of involved users and by the need to access ontology knowledge on-the-move, through mobile devices. Research challenges involved by ontology browsing and editing on mobile platforms are manifold and range from effective interaction design to effective layout on displays with reduced dimensions, to computational issues, and so on. In this paper we start addressing the above issues with a dual goal of showing the technical feasibility of the approach, and of hinting at the possible improvements brought by touch-based interaction primitives to ontology visualization and browsing experience. We therefore designed and implemented an ontology browsing application for Android exploiting state of the art interactions such as radial menus, pinch-to-zoom, etc. and focusing on touch-based interaction for mobile ontology browsers. Presented results include a working prototype application, named JellyOnt, and a first validation involving habitual users of Semantic Web technologies. User testing shows a low learning curve and positive reactions (e.g., willingness to adopt) to the proposed paradigms, which are perceived as both innovative and useful.

II. Related Works

The visualization of knowledge models has attracted, and continues to attract attention from the research community. The problem of effectively conveying information about complex models involving many aspects and based upon non-visual formalisms represents a really interesting challenge often requiring interdisciplinary approaches, e.g., by merging Semantic Web and Information Visualization. As hinted by Wang and Parsia [3], ontology visualization plays a crucial role in *sensemaking* processes when users try to grasp the domain knowledge represented by an ontology and to map such information into a mental model of the ontology itself. Requirements for ontology browsing and visualization have been widely investigated (e.g., in [1]) and include among the others the ability to provide: data overview, detailed zooming on single parts of the model, filter irrelevant data, etc. Many approaches exist, which address visualization issues by proposing different representation paradigms, each with its own pros and cons. In 2007, Katifori et al. [1] carried a systematic survey of ontology visualization approaches, organizing them according to 6 visualization categories: (1) indented list, (2) node-link and tree, (3) zoomable, (4) space-filling, (5) focus + context information, (6) 3D information landscapes. No specific method seems to be the most ap-

propriate for all applications and, as a consequence, a viable solution is to provide users with several visualizations. This approach lies at the basis of the presented work, where specific visualizations are considered interchangeable and deserving specific research attention. In more recent times, several other researches tackled the ontology visualization challenge, providing different methodologies and tools, which can somewhat be classified in the same 6 categories. Among them, Motta et al. [4] propose KC-Viz, a visualization tool that exploits an empirically-validated ontology summarization method to provide concise views of large ontologies and to support "middle-out" navigation of represented models. Hop et al., on the other hand, proposed GLOW [5], a method for ontology visualization based on Hierarchical Edge Bundles [6]: a new visually attractive technique for displaying relations in hierarchical data. Despite the active research in ontology visualization, few or no approaches exploit touch-based interactions, as in tablets or interactive surfaces. While mobile platforms have been considered viable solutions to provide semantic-powered services in mobility, e.g., as in the DBPedia mobile application [7], the adoption of touch-enabled mobile platforms for ontology browsing is almost neglected. An early attempt of providing a subset of the information stored in a given ontology model through a mobile terminal can be found in the OWL Mobile browser[1]. However, the provided interface is textual and does not exploit touch-typical interactions, unless for scrolling lists of ontology entities. In this sense, the approach proposed in this paper goes beyond the current state of the art, proposing to exploit mobile touch-based interfaces as a viable mean to improve the usability of Linked Data and of semantic-based solutions.

III. Requirements

Designing an OWL ontology browsing application for mobile devices requires to completely re-think interaction metaphors and paradigms as the typical point-and-click approach is clearly not suitable for touch devices and involved computational issues are not negligible too. We therefore tackled the application design by starting from functional and non-functional requirements.

Functional requirements involve ontology visualization and browsing operations, and are mainly extracted from approaches and surveys reported in literature (i.e., [1], [8], [9]). They encompass the following domains:

- ontology loading (local and remote);
- ontology visualization and browsing;
- ontology querying;
- full text-search on ontology entities;
- basic Create/Read/Update/Delete (CRUD) operations.

Each domain has been better specified through an iterative feature elicitation process (accounting user needs emerging from the Kriglstein survey in [8]) that led to the functional requirements summarized in Table I, covering a typical subset of ontology browsing activities.

[1]http://onto.rpi.edu/demo/owlmobile2, last visited on January 29, 2019.

TABLE I: Functional Requirements, organized by domain and priority (1 - highest, 5 - lowest).

Domain	Requirement	Description	Priority
Ontology loading	Loading from remote URLs	Load an ontology given the ontology URI.	1
	Loading from local URIs	Load a locally stored ontology.	2
	Creation from scratch	Create a new ontology from scratch.	4
Ontology visualization	Model rendering (layout)	Show a graphical representation of the loaded ontology.	1
	Concept/ Instance visualization	Support the visualization of both concepts and instances, either separately or in mixed views.	2
	Multiple layout support	Support different rendering layouts and algorithms.	3
Ontology browsing	Zoom-in/ Zoom-out	Increase the size of displayed items to tackle small display issues.	1
	Pan	Pan the visualization in any direction to show hidden elements.	1
	Bookmark	Set the current visualization as bookmark to recall between different browsing sessions.	2
	Custom view	Compose custom views including only a subset of the ontology concept and relationships.	2
Ontology querying	Execution of queries	Capability to execute queries (e.g., in SPARQL) over the visualized ontology and to integrate results in the proposed visualization.	3
	Graphical composition of queries	Ability to compose queries (e.g., in SPARQL) through graphical metaphors, focused on touch interactions.	4
Full text-search on ontology entities	Simple full text search	Full text search on ontology entities as a means to quickly navigate to specific ontology sections or to quickly specify custom views (filters).	4
Basic CRUD	CRUD functionalities	Graphical primitives to Create, Read, Update, Delete ontology entities.	3

Non Functional requirements identify constraints emerging from the target user base and the target platforms. They encompass: (a) low learning curve; (b) touch-based operations; (c) mobile platform support; (d) effective exploitation of "natural" paradigms (e.g., pinch-to-zoom, etc.); (e) responsiveness; (f) low impact on batteries; (g) open source implementation; (h) extensibility; (i) ease of use.

In this paper we focus more on requirements involving user interaction (a,b,d,i) and mobility (c) whereas remaining constraints are considered as having lower priority.

IV. Interaction design

To enable effective exploitation of "natural" interaction paradigms on tablets and touch surfaces, the application design is bounded to touch-only interaction, and tries to avoid adaptations of traditional interfaces (e.g., keyboard and mouse) as much as possible. The initial interaction design focuses

on requirements with priority level < 3 whereas low-priority issues are postponed to next design cycles.

The interaction workflow is organized into well defined *activities*, each with a specific focus and goal in terms of ontology operations. They include: (a) *welcome*, (b) ontology *browsing*, (c) ontology *querying*[2], (d) ontology *editing*[2], and (e) ontology *visualization*.

The latter can be considered as a background activity which is seamlessly combined with the others, *welcome* excluded. The general interaction workflow is depicted in Figure 1 and is deployed in two main phases: an initial selection of the model to operate on, and a subsequent execution of an ontology-related activity (i.e., browsing, querying or editing). Users can switch between activities at any time, and the transition shall always be smooth and easy to notice. Abrupt changes in the rendered interface are not allowed except for the "restart from scratch" case where the interaction workflow requires a re-start from the *welcome* activity. Even in this case transitions between activities shall be provided.

Navigation between activities, and related views, exploits *radial (or pie) menus* as core elements. Radial menus have long been investigated (e.g., see the work in [10]) and typically offer improved interaction effectiveness (e.g., by lowering the error rate) and selection speed [11] with respect to linear menus. Menu slices are large in size and easy to approach, especially on touch-enabled surfaces.

Ontology-related functions accessible through radial menus, include: (1) initial ontology selection, allowing users to select which ontology load/create, (2) activity selection, e.g., ontology browsing, querying or editing and (3) activity-related tooling, e.g., to create custom views or to bookmark specific visualizations. Depending on the specific navigation goal, pie menus can assume different sizes, shapes and locations. Three main configurations are considered: centered and full-screen for the *welcome* activity (Figure 2a), placed in a corner and covering 90° only for activity selection (Figure 2b), collapsible and spanning 360° for activity related tooling (Figure 2c).

To enable seamless switching between activities, specific radial menus can be kept visible on the application viewport, either collapsed or open. Activity selection is therefore reduced to radial menu opening/closing whereas the main area of the user interface is handled by the background *visualization*.

A. Activities

Welcome is the initial activity performed by users, where the ontology model to show is selected. It consists of a full-size, full-screen radial menu offering 2 main options: loading an existing ontology or creating a new ontology from scratch. According to well-established paradigms, the loading part (top slices) includes handling of recently opened models (Figure 2a) and offers a mean to specify the ontology to load by providing the corresponding URL.

Browsing ontology models is completely addressed in this first interaction design cycle, as it tackles most of the identified mid-high priority requirements. Ontology browsing is a

[2]not specified here as pertaining to low-priority requirements.

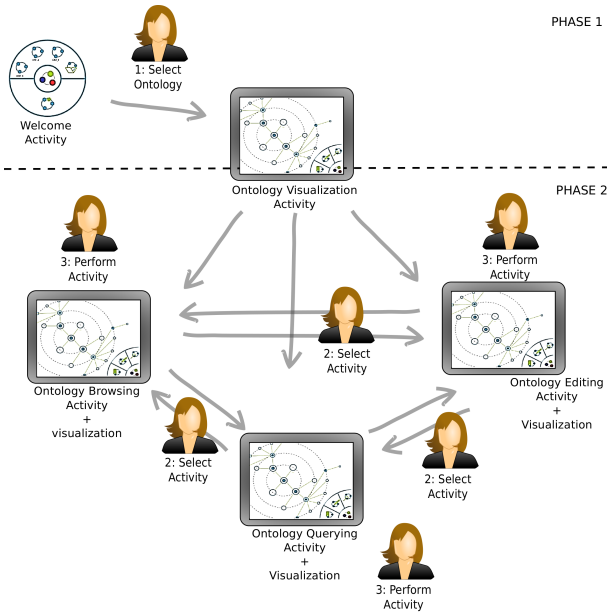

Fig. 1: Interaction Workflow

composite activity where the background visualization carries typical viewport navigation tasks, e.g., panning and zooming, and is complemented by a set of interaction paradigms specifically designed to support navigation of ontology models: bookmarking and custom views and exploit well established patterns stemming from the Web domain (*bookmarks*) and from the Computer Aided Design domain (*custom views*). Each pattern is initiated through a tap on one slice of the specific radial menu associated to the browsing activity.

Bookmarking allows users to name a specific visualization of the ontology (zoom-level, position, etc.) for later recall. Bookmarks are ontology-specific and persistent; each ontology URI defines a different bookmark namespace and same-named elements might exist for different ontologies. Bookmark persistence allows survival of named visualizations over different browsing sessions and over device deactivation. Bookmarks can be promptly retrieved through a bookmark selection pop-up. During the browsing activity, users can specify *custom views*, i.e., filtered visualizations that only render a selected subset of ontology concepts and relationships. While detailed implementation of the rendering logic is demanded to the visualization activity, and to the specific layout algorithm adopted, involved concepts and relationships are defined through the custom view menu entry and persistently stored to be later recalled. Similarly to bookmarks, views are ontology-specific (same-named views for different ontologies might exist) and survive to different browsing sessions and device deactivation.

Querying and Editing have only been partially tackled. For SPARQL querying we envisioned a graphical query composition pattern similar to GQL [12]. For what concerns ontology editing, a preliminary wireframe design has been completed, exploiting "natural" touch interactions with typical text-filling tasks related to ontology creation (e.g., concept and relation

(a) Welcome activity (b) Activity selection (c) Activity-specific tooling

Fig. 2: Different radial menus configurations adopted in JellyOnt.

naming) that exploit keyboard-like patterns.

Visualization is the central activity of any ontology viewer as it conveys information to users by means of suitable graphical representation of ontology entities (concepts, relationships, instances, etc.). Defining how an ontology shall be represented on a screen and what kind of navigation metaphors need to be adopted for diving into the represented knowledge is subject of many research efforts both in the Semantic Web and in the Information Visualization domains. In this paper we explicitly avoid to tackle the representation issue, which deserves specific research efforts (as shown in literature), and, instead, we focus on touch-based interaction for ontology browsing. In this sense, "pure" visualization concerns are of lower priority with respect to interaction elements presented in previous sections even if we are perfectly aware that different visualization choices might have a not negligible impact on the overall user experience. For the sake of simplicity we therefore limit visualization to a very simple layout and navigation paradigm, radial tree (Figure 4), while concentrating on a software infrastructure capable of supporting many visualization solutions, thus enabling further investigation on this theme in a touch-based mobile environment.

V. JellyOnt

We applied the designed interactions to JellyOnt, an ontology browsing application for Android. In this first prototype we choose not to implement the query and the editing activities, since they pertain to low-priority requirements.

JellyOnt exploits a layered architecture (Figure 3) aimed at decoupling ontology access primitives from graph representation engines and from node-layout algorithms. At the lower layer, ontology access is handled by full-blown, well established ontology APIs such as Apache Jena [3] or OWL API[4]. The ontology model exposed by such APIs is then abstracted into a shared graph representation, designed taking inspiration from the Prefuse Information Visualization Toolkit[5] architecture. This shared representation offers standard query primitives to layout algorithms, thus enabling them to get

[3]https://jena.apache.org, last visited on January 29, 2019
[4]http://owlcs.github.io/owlapi/, last visited on January 29, 2019
[5]https://github.com/prefuse/Prefuse, last visited on December 17, 2018

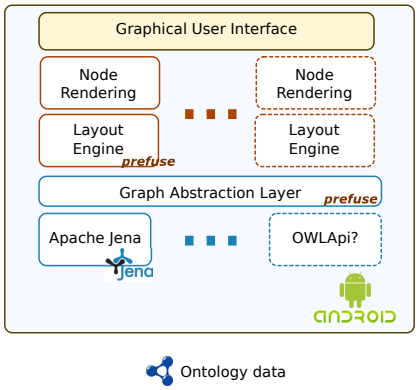

Fig. 3: The JellyOnt logic architecture.

the information needed to successfully set-up an ontology visualization.

In the current JellyOnt implementation a RadialTree layout engine has been implemented by "porting" the corresponding Prefuse layout. The same operation can be done for other layouts either available in Prefuse or stemming from other sources, e.g., algorithms recently defined in literature [4], [13].

Each layout engine can exploit custom-node renderers (the "Node Rendering" modules reported in Figure 3) to assign a visual representation to the node positioned on the application viewport. Node renderers can be changed at runtime, thus enabling advanced interface building (implementing, for example, node overlays, animations, etc.)

The actual implementation of JellyOnt maps the activities identified in the design process to real Android activities, and manages switching between them as defined in the workflow design. It targets an Android 4.x environment (or later) and has been tested on different platforms including some Asus Transformer Pad (TF300T/TF300TG) and Samsung Galaxy Tab 10.1 tablets (Figure 4 shows a screenshot of the application.

Several technical challenges have been addressed in the application development encompassing: (a) radial menu development, (b) porting/adaptation of ontology APIs, (c) porting/adaptation of the Prefuse layout management architecture. In particular, ontology access and elaboration has been per-

TABLE II: The nine tasks used for the study

Task	Description
T1(a)	Open the http://www.co-ode.org/ontologies/pizza/pizza.owl ontology
T1(b)	Open the most recent ontology
T2	Open the menu responsible for storing any ontology characteristics
T3	Open and move a menu at your choice
T4	If no menu is open, open and then close one of theme
T5	Move the visualized ontology on the screen and save a new bookmark
T6	By using a menu, store the root node and the leaf node in a new view
T7	Open an existing bookmark
T8	Open and then close an existing view
T9(a)	By using a menu, delete an existing bookmark
T9(b)	By using a menu, delete an existing view

formed using the full-blown Apache Jena framework, adopted to run on Android devices. Ontology loading performances proved to be sufficient to support effective interaction, with acceptable loading times in the order of few seconds (obviously the loading time depends on the specific ontology size and upon the available network bandwidth).

VI. Evaluation

A preliminary user evaluation has been carried on to identify the relative strengths and weaknesses of JellyOnt, to roughly estimate the ability to use the app without external hints, and to check whether the functionality offered by the pie menus are easy to discover and use. Four participants tried the mobile application in a controlled environment, performing 9 tasks each, and replying to a final questionnaire. Tasks have been inspired by requirements previously reported, while participants observations allowed a qualitative analysis of the proposed interaction paradigm and interface, to help identifying future directions. The four participants recruited for this study were 2 females and 2 males, aged 26 to 46 (with an average age of 34). Participants were selected with similar skill level, especially about usage and knowledge of Semantic Web technologies in general, and OWL ontologies in particular.

All worked in technology-related fields; two of them were Ph.D. students in Computer Engineering, while the other were researchers. The study was held in Italian and the interface was localized accordingly. The platform chosen for the test was a 10" tablet. Each participant performed each task in counterbalanced order, to reduce the chances that the order could influence the final results.

After a short introduction to the study and collection of demographic data, participants were free to test JellyOnt for a few minutes, to form a preliminary opinion about the application. Each user was told to complete a set of nine tasks, one at a time. Examples of proposed tasks include: "Open the most recent ontology" and "By using a menu, store the root node and a leaf node in a new view". Each participant received a different version of task 1 and 9 (see Table II for the complete task list), to evaluate similar functions but without duplicating any task. At the end, participants were given a questionnaire and asked to rate their agreement with some sentences about the look of the application, its learning curve and the overall layout. Users open comments and explanations were collected (e.g., problems found or explanation about something done during a task) during debriefing interviews. The duration of the entire study ranged between 10 and 20 minutes. Every user completed with success every task, without any major problem.

VII. Results and Discussion

The *success rate* of each participant, i.e., the percentage of tasks that users completed correctly, was very high. Every user completed with success every task, except task 4: nobody was able to close a visible activity-specific pie menu. The problem was due to the absence of any visual feedback on the prototype application: menu slices, in fact, do not have a "selected" state and they appear identical when the corresponding activity-specific menu is visible or hidden. The final questionnaire asked participants to express their agreement about four sentences: a) I like the appearance of the application. b) I think that the app is intuitive. c) I think that the overall layout is understandable. d) It is easy to learn how to use the application.

Results from participants were satisfying: most users agree or strongly agree with the proposed sentences. Only one participant did not express a preference about the third sentence, due to some difficulties in understanding what functionality is associated to some icons. During the debriefing interview, we collected some observations from the participants, about their behavior during the study and about what works in the application. All the participants found the radial menu in the *welcome* activity very intuitive. After opening the ontology, each user tries to tap over a node for executing the second task, with the desire to see further information or to perform some actions on the tapped node (supported by the JellyOnt architecture but not yet implemented). According to this behavior, they suggest to add pertinent actions to each node. Most users found quite difficult the *bookmarks* and *views* management; however, they noticed that such tools are useful and that their problems could be avoided with some help or with a longer usage of the application. The choice made for the view visualization, i.e., to show everything selected up to the root node, was particularly appreciated. Furthermore, participants suggest to introduce a "back" button to revert any action done on the visualized ontology, such as returning to the default ontology view when showing a bookmark.

Nobody had any difficulty with the implemented gestures: users had panned and zoomed the ontology with an extreme naturalness. In the same way, most icons present in the application were immediately understood; the only exception was the "browse" icon, probably to be replaced. Finally, according to most users, the application lacks of the possibility to change the visualization layout. This aspect, in fact, was

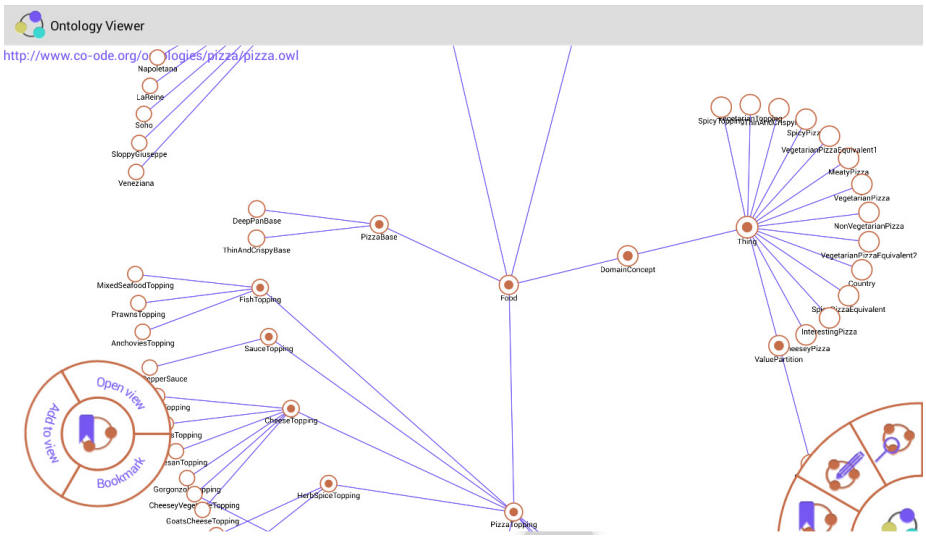

Fig. 4: A JellyOnt snapshot (colors have been inverted for the sake of readability).

present during the requirements and design phases but omitted in this first working prototype. All the participants, before concluding the debriefing interview, expressed appreciation on the idea, and the desire to be aware of further investigations and experiments on the same topic. Overall, the evaluation of the prototype was positive and provided useful insight on touch-based ontology browsing applications. It confirms, also, most of the design choices we made and offers precious hints for continuing the development and the refinement of the application.

VIII. Conclusions

This paper presented the design of a touch-based ontology browser for effective and highly appealing presentation of new OWL ontologies, novel modeling approaches, and experiments with different visualization layouts in mobility. Formal and non formal requirements, extracted from the literature, guided the design phase, while a first working prototype, realized as an Android app named JellyOnt, has been described and evaluated by means of an initial user study. Experimental results are encouraging: all users involved in the test enjoyed JellyOnt and provided useful insights for continuing the development and the refinement of the application. Future works will complete the design and implementation of the querying and editing activities and will extend support for multiple visualization by implementing well-known layout and rendering algorithms. Possible extensions of the approach will ultimately include new visualization paradigms and possible applications to larger interactive surfaces (e.g., touch-based tabletops).

Acknowledgements

The authors wish to thank Alessandra Ronsini for the initial development of the prototype application, Dario Bonino for his support and consultancy about ontologies, and the anonymous volunteers involved in the evaluation phase.

References

[1] A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou, "Ontology visualization methods - a survey," *ACM Comput. Surv.*, vol. 39, no. 4, Nov. 2007.

[2] M. Dzbor, E. Motta, C. Buil, J. Gomez, O. Görlitz, and H. Lewen, "Developing ontologies in OWL: An observational study 1," CEUR Workshop Proceedings, 2006.

[3] T. D. Wang and B. Parsia, "CropCircles: topology sensitive visualization of OWL class hierarchies," in *Proceedings of the 5th international conference on The Semantic Web*, ser. ISWC'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 695–708.

[4] E. Motta, P. Mulholland, S. Peroni, M. d'Aquin, J. M. Gomez-Perez, V. Mendez, and F. Zablith, "A novel approach to visualizing and navigating ontologies," in *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ser. ISWC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 470–486.

[5] W. Hop, S. de Ridder, F. Frasincar, and F. Hogenboom, "Using hierarchical edge bundles to visualize complex ontologies in GLOW," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 304–311.

[6] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 12, no. 5, pp. 741–748, 2006.

[7] C. Becker and C. Bizer, "DBpedia mobile: A location-enabled linked data browser," Semantic Web Challenge, 2008.

[8] S. Kriglstein, "User requirements analysis on ontology visualization," in *Complex, Intelligent and Software Intensive Systems, 2009. CISIS '09. International Conference on*, 2009, pp. 694–699.

[9] A.-S. Dadzie and M. Rowe, "Approaches to Visualising Linked Data: A Survey," *Semantic Web*, vol. 2, no. 2, pp. 89 – 124, 2011.

[10] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman, "An empirical comparison of pie vs. linear menus," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '88. New York, NY, USA: ACM, 1988, pp. 95–100.

[11] K. Samp and S. Decker, "Visual search in radial menus," in *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV*, ser. INTERACT'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 248–255.

[12] M. Zviedris and G. Barzdins, "ViziQuer: A tool to explore and query SPARQL endpoints," in *The Semanic Web: Research and Applications*, ser. Lecture Notes in Computer Science, G. Antoniou, M. Grobelnik, E. Simperl, B. Parsia, D. Plexousakis, P. Leenheer, and J. Pan, Eds. Springer Berlin Heidelberg, 2011, vol. 6644, pp. 441–445.

[13] M. Lanzenberger, J. Sampson, and M. Rester, "Ontology visualization: Tools and techniques for visual representation of semi-structured metadata," *Journal of Universal Computer Science*, vol. 16, no. 7, pp. 1036–1054, apr 2010.