April 2017

Multi-View Ontology Explorer (MOE): Interactive Visual Exploration of Ontologies

Zhao Lin
The University of Western Ontario

Supervisor
Dr. Kamran Sedig
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Abstract

An ontology is an explicit specification of a conceptualization. This specification consists of a common vocabulary and information structure of a domain. Ontologies have applications in many fields to semantically link information in a standardized manner. In these fields, it is often crucial for both expert and non-expert users to quickly grasp the contents of an ontology; and to achieve this, many ontology tools implement visualization components. There are many past works on ontology visualization, and most of these tools are adapted from tree and graph based visualization techniques (e.g. treemaps, node-link graphs, and 3D interfaces). However, due to the enormous size of ontologies, these existing tools have their own shortcomings when dealing information overload, usually resulting in clutter and occlusion on the screen. In this thesis, we propose a set of novel visualizations and interactions to visualize very large ontologies. We design 5 dynamically linked visualizations that focus on a different level of abstraction individually. These different levels of abstraction start from a high-level overview down to a low-level entity. In addition, these visualizations collectively visualize landmarks, routes, and survey knowledge to support the formation of mental models. Search and save features are implemented to support on-demand and guided exploration. Finally, we implement our design as a web application.

Keywords: Visualization, Interaction, Exploration, Navigation, Ontology

Acknowledgements

I would like to express my deepest gratitude for my supervisor Dr. Kamran Sedig, who have provided me with ongoing support and constructive criticism on this thesis. I would also like to thank Dr. Paul Parsons and colleague Jon Demelo for their valuable feedbacks and advice.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Motivation

With advances in computer technologies there has been a growing dissemination of huge data sets in various disciplines and industries. Across domains of information, data is being accessed, modified and interchanged among users and software applications. A growing challenge for systems dealing with large shared or reused data sets is keeping a consistent set of vocabulary and understanding the underlying meaning of the data—the *semantics*. For instance, in the field of biomedicine, if a phenotype (an observable characteristic of an individual) is transferred from one software agent to another, how can we systematically ensure that it is the intended phenotype across all the software agents? Given two phenotypes from different applications, how can we compare their semantic relationships? that is, is one a more general term of the other? or are they the same phenotype but with a different spelling? An approach first introduced in the field of Artificial Intelligence, and later popularized with the Semantic Web, is to employ domain specific meta-data, or an ontology, to formally represent a domain of knowledge through explicit entities and relationships.

More formally, an *ontology*[1] is defined as: "an explicit specification of a conceptualization" (Gruber, 1993). An ontology defines a common vocabulary for a domain of knowledge. Among others, an ontology enables sharing of the structure of information; enables reuse of domain knowledge; and make domain assumptions explicit (Noy & McGuinness, 2001). An ontology contains classes, properties, and relations that formally describe a domain of discourse. Each class represents an entity in the domain. Each class contain properties that describe features of the class. A set of *is-a* relations define the inheritance relationship among the classes. In addition, class properties can contain object relations that connect classes through domain specific relationships. For instance, in the foundational Model of Anatomy ontology (FMA)[2], the *part-of* relation describes containment among classes. More specifically, an ontology is a triple $O = \{C, R, isa\}$ where:

[1] The term ontology is borrowed from Philosophy, meaning a systematic account of Existence.
[2] http://www.ontobee.org/ontology/fma

1. $C = \{c_1, c_2, c_3, \ldots c_n\}$ is the set of classes;
2. $S = \{s_1, s_2, s_3 \ldots s_n\}$ is the set of slots (properties) or binary roles/relations among classes; and
3. *isa*, the set of inheritance relations.

Further, a set of instances $i = \{i_1, i_2, i_3 \ldots i_n\}$ where i_w is an instance of class c_w with concrete values for each property of the class (Amann & Fundulaki, 1999). As an example, Figure 1 depicts a subset of the wine ontology[3] represented in a UML diagram. Each box is a class, and each arrow represents an inheritance relationship. Here, each wine has the properties color, maker, and location. Applications that adopt the use of this wine ontology can keep a consistent set of vocabulary and semantic relationships. For instance, both *IceWine* and *SweetRiesling* is semantically related, as they are both a type of *DessertWine*. Communications in wine data among applications using this ontology would therefore be less ambiguous and systematic.

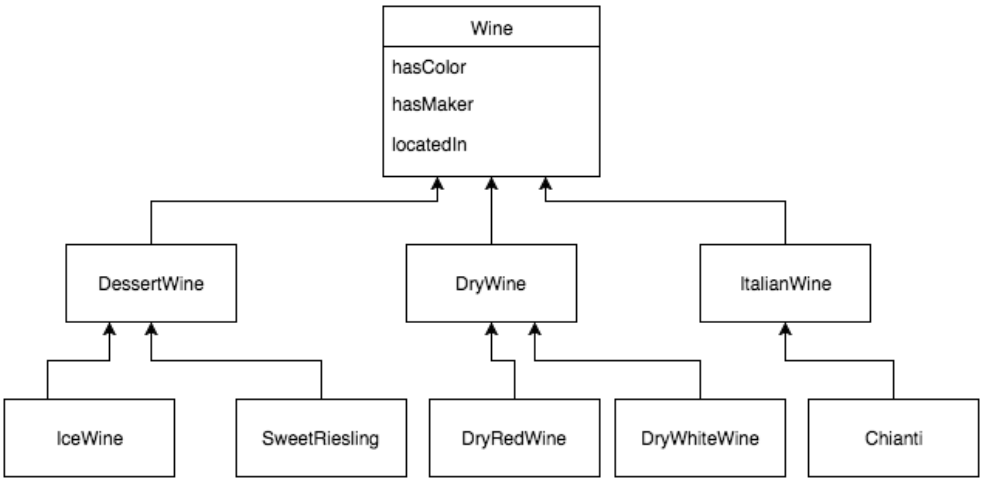


Figure 1. A subset of the Wine Ontology

[3] https://www.w3.org/TR/owl-guide/wine.rdf

1.2 Applications of Ontologies

Ontologies are used in in almost every domain and industry, both academic and commercial, including medicine, chemistry, law, business, and engineering, among others. In life sciences, an ontology has applications in 3D modeling and visualization. For instance, in the field of embryology, the discipline concerning morphologic changes of organisms, it contains a high level of complexity concerning physiological and pathological mechanisms. To address the complexity, computer modeling and simulation are used to assist and visualize the information. In addition, the ontology My Corporis Fabrica Embryo (MyCF) (Rabattu et al., 2015) is developed to keep the 3D models consistent with the knowledge base. MyCF, among others, aims to make explicit links between anatomical entities, human body functions, and 3D models. In the absence of the link among the 3D models and the anatomical ontologies, it is difficult and time-consuming to access the anatomical content from the 3D models and vice versa.

Another purpose of ontologies is rooted in the Semantic Web initiative. The Semantic Web aims to shift the World Wide Web in the direction of *linked-data.* Historically, web pages are linked from pages to pages by an HTML hyperlink. However, the with the explosion of data in every domain, it is now beneficial to link the data itself from resources to resources. Hence, the term *linked-data* links the underlying data and its semantics across the sea of information to form the *web of data* (WOA) (Bizer et al., 2008). Much like an URL (Uniform Resource Locator) for a webpage, data resources have an URI (Uniform Resource Identifier) to uniquely identify itself. Resource data that are linked allow users to navigate from one data source to another that share connected information more efficiently. For example, suppose we have two databases: one containing geographic information including geographic locations and their rock formations, and another database regarding rocks. In this scenario, *linked-data* aims to connect the two databases via a common knowledge (i.e., rocks) to allow a systematic linking of geography and rocks data.

An example of using ontologies in linked databases on the web can be seen in marine biology in the works of (Dhillon et al., 2013). In their work, Dhillon et al. investigated in creating a digital biological ecosystem of fish. Such digital systems can help solve many business problems. In creating the digital biological environment, a series of databases are incorporated: geographic information systems, fish species ontology, map contours, and fish distribution data from scientists.

Their work includes a visualization tool that connects fish distribution data with map contours on to a geographic map representation. These databases are 'linked' in the sense that users can analyze the fish distribution data from the scientists while seeing their connections with other associative data sources, i.e. map contours, or detailed fish information.

In describing the ontology for machine reading, the W3C [4] recommends RDFs (Resource Description Framework Schema) and OWL (Web Ontology Language) file formats. Another format of ontologies is OBO, which is more prominent in the biomedical community. Both RDFs and OWL is a data model in XML describing declarative data statements. Each statement is a triple of the form object-attribute-value (Antoniou, Groth, Van Harmelen, & Hoekstra, 2012). As an example, in describing the wine ontology, we can compose it from a set of statements, and once such statement is: "*IceWine* is a subclass of *DessertWine"*. Where the *IceWine* class is the object, subclass of is an attribute, and *DessertWine* is the value. Indeed, there are much more technicalities in the specifications of an ontology file such as property value ranges, property types, and logical inferences among classes. A more in depth introduction to the construction of such files can be found in (Antoniou et al., 2012).

1.3 Problem Background

The process of creating and maintaining an ontology is in the field of ontology engineering (Suárez-Figueroa, García-Castro, Villazón Terrazas, & Gómez-Pérez, 2011). Here, maintainers are concerned with the life cycle of an ontology using methodologies, tools, and languages to implement the ontology (e.g. RDFS or OWL). A few prominent ontology editors include Protégé[5] (Noy et al., 2001), Apollo[6], and NeOn[7] (Suárez-Figueroa, Gómez-Pérez, & Fernández-López, 2012). However, ontology sizes can get extremely large, and it may be difficult for maintainers and users to grasp the ontology contents and effectively complete their tasks. For instance, in the

[4] The World Wide Web Consortium (W3C) is an international community working to develop Web Standards.
[5] http://protege.stanford.edu/
[6] http://apollo.open.ac.uk/
[7] http://neon-toolkit.org/wiki/Main_Page.html

Human Phenotype Ontology[8] (HPO), there is a total of over 11,000 terms, and in the Gene Ontology[9], there are over 47 thousand terms[10]. In short, ontologies can be huge in scale with heterogeneous contents and have a very complicated structure to understand.

An approach to assist in dealing with this complexity is with computer based interactive visualization tools. There has been numerous past works on ontology visualizations. Because ontologies take form of an acyclic[11] directed graph, many graph visualization techniques have been applied (e.g. node-link graphs). In addition, because there is a hierarchical nature through the *is-a* inheritance relationships, many tree visualizations techniques have been applied (e.g. treemaps). Nonetheless, the huge complexity of an ontology is astronomical compared to a computer screen real-estate. In effect, many ontology visualization tools suffer from over-crowding and occlusion of data.

1.4 Problem Statement:

A moderately large ontology (>10,000 entities) can be difficult to visualize effectively with traditional methods. If abstraction techniques are avoided, an attempt the visualize the whole ontology is futile given the limited screen space. In addition, by using traditional methods that temporarily hide specific areas of the ontology and reshowing them again on demand (via interaction) is susceptible to the loss of surrounding context. Finally, many ontology visualization tools lack the facilities to support the formation of mental models, that is, many tools are inconsistent with the encoding of key landmarks, routes, and survey knowledge formation. Given these considerations, we present a prototype of an interactive visualization tool for ontologies with the following goals:

1. Support the visualization of very large ontologies;
2. Support the rapid exploration and navigation of ontologies; and
3. Support sense-making in the formation of mental models of ontologies.

---

[8] http://human-phenotype-ontology.github.io/
[9] http://www.geneontology.org/
[10] These counts are determined from Protégé's Ontology metrics panel
[11] Cycles in an inheritance relation tree would infer that all the connected terms are equivalent

1.5 Approach

In supporting the task of sense-making, we use techniques from the literature of information visualization, visual analytics, and human-computer interaction to develop a prototype of a novel interactive visualization tool. Indeed, visualization techniques have been shown to support the exploration of large data sets (Keim, 2002). Traditional software algorithms are very good at solving well-defined problems that concerns with only the computer (e.g. finding the shortest path in a graph). However, here, the problem is much more ill-defined as we are concerned with continuous interactions between the user and the tool. Therefore, we need to take into consideration the ease of use, user interface design, representation techniques, and interaction techniques, among others, to best accommodate the formation of a user's internal mental model.

To address the complexity of the ontology, we take a multi-view approach to visual exploration. That is, a single representation is generally not enough to show every perspective of an ontology's structure. This approach uses multiple coordinated visual representations that focus on different levels of abstraction. These different levels of abstractions start from a high-level overview down to a low-level entity[12]. In addition, we employ a metaphor consistent with ideas in the formation of spatial mental models (Tversky, 1991). That is, much like forming a mental model of a physical space, we take consideration in encoding key *landmarks*, *routes* or paths from entity to entity, and *surveys knowledge*, or a general overview. In result, the prototype aims to effectively visualize very large ontologies and provide interactions to allow incremental navigation and exploration of the ontology information space.

1.6 Thesis Outline

This thesis is organized as follows: **Chapter 2** is a review of previous related ontology visualization tools with an analysis on their advantages and disadvantages. **Chapter 3** reviews background on visualization design methodologies and interaction techniques used in building the

[12] Henceforth, entity, class, and node may be used interchangeably to refer to an entity or class in the ontology.

prototype. **Chapter 4** describes the prototype in detail. And **Chapter 5** concludes the thesis with discussion and future work.

Chapter 2 Ontology Visualization Tools

2.1 Survey of Past Ontology Visualization Tools

In literature, there is a numerous number of past work on ontology visualization. In previous years, many visualization methods and representation styles originally designed for tree and graph based structures has been adopted for ontologies, and in more recent years, more sophisticated interaction and distortion techniques has been adopted to deal with the complexity of ontologies. In survey works such as (Katifori, Halatsis, Lepouras, Vassilakis, & Giannopoulou, 2007), (Wang & Almeida, 2007), (Sagha, 2014), and (Bikakis & Sellis, 2016), a list of past and state-of-the-art ontology visualization methods and tools are described and compared. Here, we present a summary of the survey works' findings.

Katifori et al. groups the existing visualization tools into six groups, namely, they are indented list, node-link and tree, zoomable, space-filling, focus+context or distortion, and 3D information landscapes (Katifori et al., 2007; Katifori, Torou, Vassilakis, Lepouras, & Halatsis, 2008). These groups differ primarily based on their representation, presentation, and interaction techniques. A tool can be in one or more groups, e.g. a treemap is both space-filling and zoomable. The following outlines the key groups in more detail and discuss their advantages and disadvantages.

2.1.1 Indented List

The indented list is a visualization style that is commonly found in file managers such as Microsoft Windows' File Explorer or Apple Macintosh's Finder application. It is also a common baseline display for hierarchies in ontology editors such as Protégé (shown in Figure 2) or Kaon2[13]. In this visualization style, the children of a node are slightly indented below the parent, and in effect, all nodes on the same level will have the same amount of indentation. Browsing the ontology, i.e. traversing downwards, involves clicking the arrow beside the parent node, thus expanding the children. In many evaluations (Katifori et al., 2007; Lanzenberger, Sampson, & Rester, 2010), this simple technique outperforms many other visualization styles in terms of clarity and ease of use.

[13] http://kaon2.semanticweb.org/

As shown in Figure 2, the full name of the ontology class name is displayed showing no occlusion. In addition, there is a clear indication of hierarchy from the indentation. However, this representation of hierarchy has its limitations with ontologies. Because it is displaying a strict tree hierarchy, nodes that appear under multiple parents (i.e. multiple inheritance) are duplicated in all the respective parents, and in result, the graph-like nature of inheritance is hard to see. In addition, if a user traverses very deep into an ontology, the relationships among expanded siblings can become unclear. Further, a deep nested tree can lose the original context.

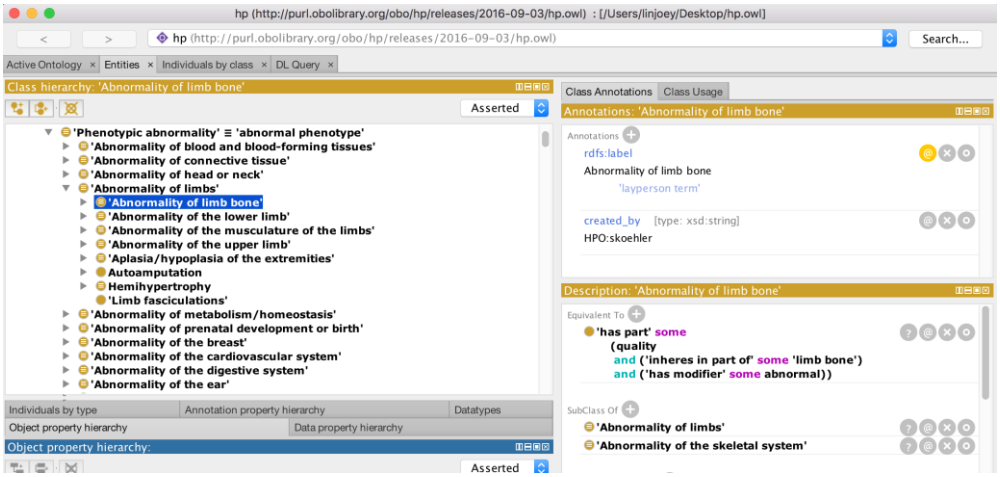


Figure 2 Protégé Class Browser

2.1.2 node-link

The node-link representation style borrows ideas from the traditional visualization techniques for graphs. Here, classes are displayed as circular or square nodes connected by lines (or links). In this representation style, parentage (including multiple inheritance) is explicitly displayed. Many ontology visualization tools adopt this approach as their representation due to its clear representation of relationships. In Figure 3, the VOWL[14] web visualization tool adopts this style. However, this representation style has many limitations in visualizing very large ontologies. There is a very inefficient use of space: the space between nodes and links are never used. As seen in

[14] http://vowl.visualdataweb.org/webvowl.html

Figure 3, only a handful of terms are displayed clearly. In attempting to display hundreds or thousands of nodes, the computer screen would quickly clutter and occlude information.

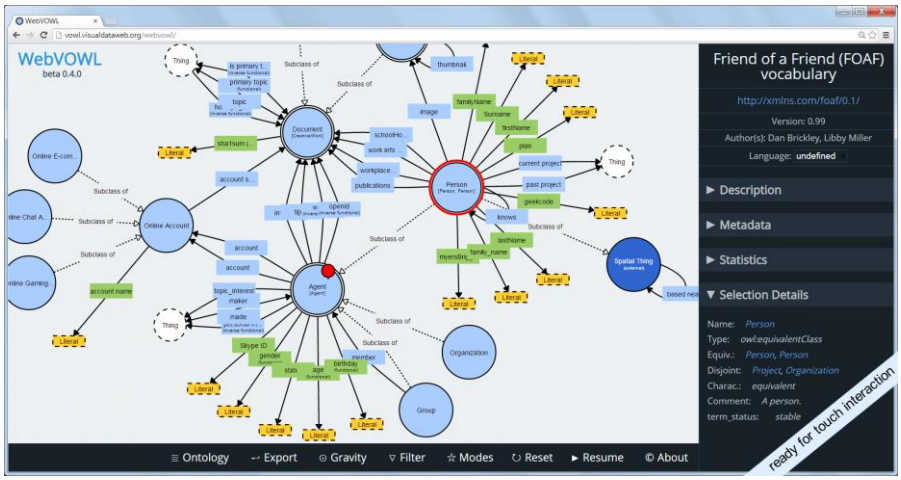


Figure 3 VOWL ontology visualization

2.1.3 zoomable

In dealing with *information overload* (Meyer, 2013), i.e. having to process and understand too much information at once, visualization systems have adopted *interaction* techniques to progressive show or hide information; one of such techniques is zooming or *drilling.* The zooming capability is available in VOWL: scrolling up or down on a mouse moves the visualization progressively larger with less information or smaller with more information, respectively. However, zooming too much or too little can have negative consequences. For instance, if a view is zoomed too much, it can lose the global context. And if view is zoomed too little, clutter can appear and occlude information.

2.1.4 space-filling

The space-filling technique tackles wasted space, as seen in node-link visualizations, by using the entire screen and subdividing sections to each child. This subdivision of space can be based on any numerical property values of a node or simply the number of children of a node. A representation of hierarchy that uses space-filling is treemaps. In Figure 4, the phenotype ontology is visualized

in a zoomable treemap[15]. Here, each subdivision size is proportionate to a node's total connected children, that is, the larger squares represent nodes with more children. The treemap has been shown to display attributes of leaf nodes effectively (Johnson, 1992), this is because color encoding on each subdivision based on a node's property value can be shown clearly without clutter. However, in a treemap, hierarchical information is very hard to decipher. Despite attempts to mitigate this issue, such as in the works of cushion treemaps (Van Wijk & Van de Wetering, 1999), specific object or inheritance relationships of an ontology is very difficult to see.

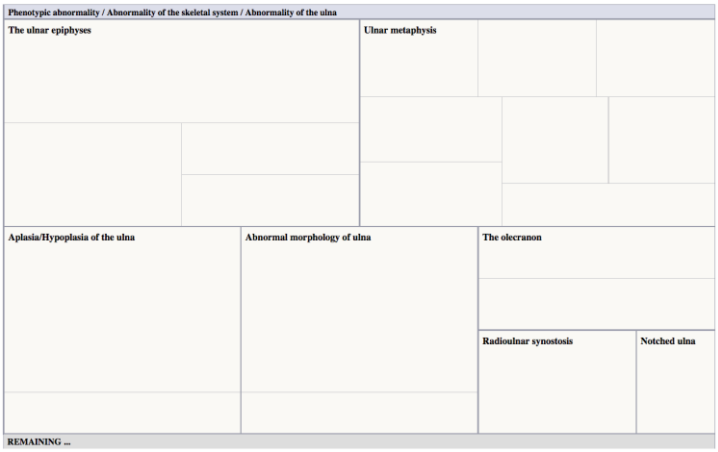


Figure 4 Treemap visualization

2.1.5 focus+context or distortion

A drawback of zoomable interfaces is that the surrounding context can be lost. In visualization tools such as node-link graphs or space-filling treemaps, we can lose sense of a node's surrounding siblings or parentage when then interface is zoomed. A technique to mitigate this issue and maintain the context is through distortion or focus+context. An example of this is done in the hyperbolic tree visualization (Lamping, Rao, & Pirolli, 1995), seen in Figure 5. In this technique, the node of interest is enlarged on the centered of the screen while surrounding nodes (the context) is shrunken on the edges, much like if the nodes were placed on a hyperbolic space. However, an

[15] http://linjoey.github.io/treemap-vis/

issue concerning this visualization is, during navigation, the constant change of moving nodes to the center can be disorienting and result in the loss of global overview.

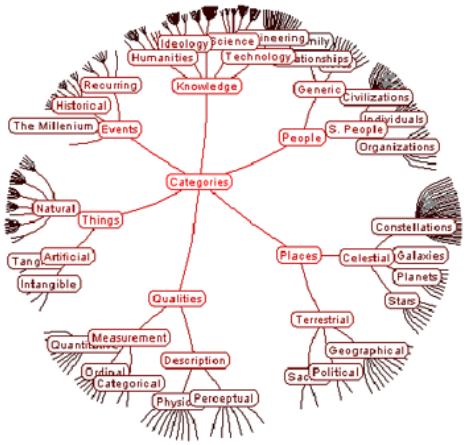


Figure 5 Hyperbolic tree visualization

2.1.6 3D

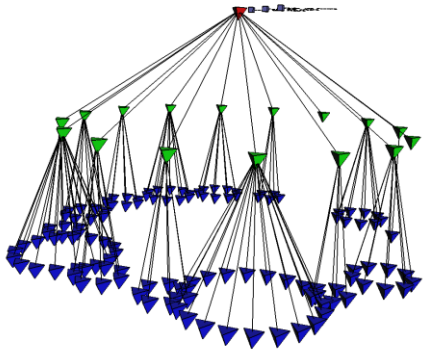


Figure 6, 3D Cone Tree visualization

The last group of ontology visualization styles concerns with using 3D objects as the representation. Many traditionally 2D visualizations such as treemaps, node-link graphs, or hyperbolic trees can be implemented in the 3D space. In particular, the Cone Tree (Robertson, Goerge G.; Mackinlay, Jock D.; Card, 1991), as seen in Figure 6, visualizes each subtree as a 3D cone that connects to the root. 3D visualizations like the Cone Tree take advantage of the 3rd dimension and this allows them to fit more content into less space. However, this also means more complicated interaction and

animation techniques are required. In addition, evaluations (Katifori et al., 2007) have shown that 3D visualizations are generally problematic for novice users—usually suffering loss of context when zooming.

2.2 Contemporary Ontology Visualization

The ontology visualization methods outlined above mostly pertain to works prior to 2007. However, each style has its own advantages and disadvantages, and not one is particularly stronger than the other in terms of the overall visualization goals outlined in this thesis. That is, each visualization style has a strong advantage in one area (e.g. node-link graphs in showing relationships), but also lack in other areas (e.g. zoomable treemaps losing context). It is clear that these visualization styles alone are insufficient in dealing with large ontologies. More recent work attempts to address this issue and we outline them below.

2.2.1 KC-Viz

KC-Viz (Motta, 2012) is a more recent approach to visualizing ontologies. Here, the authors are concerned with the sense-making tasks involved in exploring ontologies. The importance of supporting tasks related to understanding an ontology's structure or a 'global' model of the ontology is emphasized. In their work, an ontology is first preprocessed through a summarization algorithm to determine the nodes of most importance. This network of '*key concepts'* is then displayed in a node-link graph representation as shown in Figure 7. This starting point of *key concepts* then support a 'middle-out' approach to exploration. KC-Viz supports interactions such as zooming and history keeping to support the exploration tasks. In addition, each subtree that are hidden is indicated in a green arrow. Further, the sizes of a hidden subtree are displayed in brackets with two numbers, one indicating the number of immediate children, and the other indicating the number of total children. Although the authors have not yet conducted an evaluation on KC-Viz, it may suffer similar issues pertaining to node-link representations; that is, a very limited number of nodes can be displayed on the screen before the context or overview is lost through clutter or occlusion.

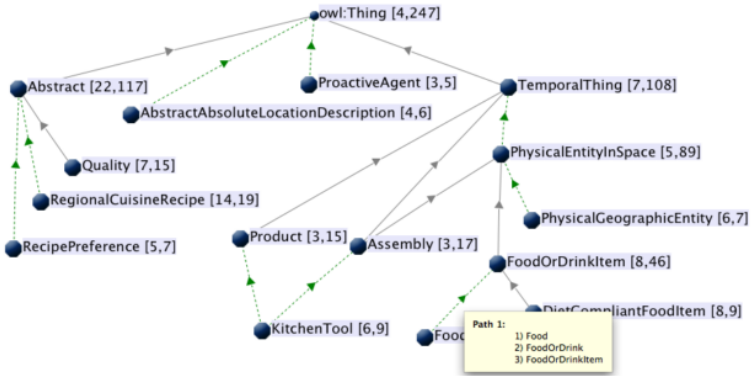


Figure 7 KC-Viz ontology visualization

2.2.2 OntoTrix

Another recent approach in visualizing ontologies is OntoTrix (Bach et al., 2011) (Figure 8). In OntoTrix, the authors are more concerned with visualizing the instances of the ontology, or a *populated* ontology. OntoTrix takes inspiration from a previous visualization technique known as NodeTrix (Henry, Fekete, & Mcguffin, 2007). Here, OntoTrix uses a style of hybrid visualization using both node-link and adjacency matrix representations. According to the authors in (Bach et al., 2011; Henry et al., 2007), this style is very efficient at visualizing locally dense but globally sparse networks. That is, each matrix focus on dense subgraphs while the matrices are connected via links providing global overview. In addition to the NodeTrix style representation (Figure 8 A), OntoTrix provides multiple coordinated views that show a different aspect of the ontology. That is, Figure 8 (B) provides an overview, Figure 8.C provides a class hierarchy, and Figure 8 (D) provides a hierarchy of class properties. Overall, OntoTrix conforms to the information-seeking mantra originally proposed by Shneiderman (Shneiderman, 1996), that is, overview first, zoom and filter, and details on demand. Although each matrix aims to handle dense subgraphs efficiently, it may still face challenges with extremely large ontologies. Here, the authors visualized the NTN ontology[16] which contains only 49 classes and 724 instances. In the case of ontologies with tens

[16] http://www.semanticbible.com/ntn/ntn-view.html

of thousands of classes or instances, each matrix would potentially still have the problem of scalability leading to clutter and occlusion.

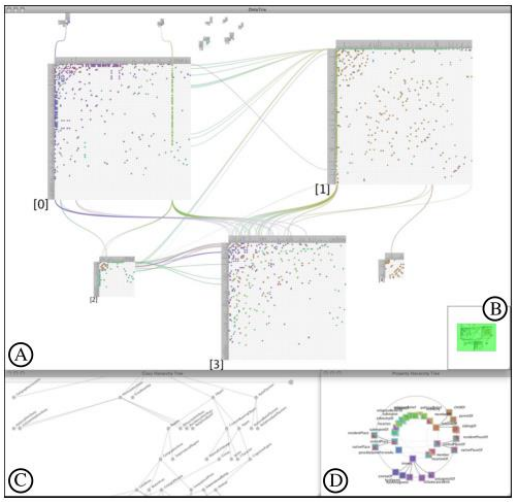


Figure 8 OntoTrix visualization. (A) main NodeTrix view (B) Bird's eye view, (C) Class hierarchy view, (D) property hierarchy view

Chapter 3 Information Visualization Background

In previous chapters, terms such as visualization, representation, presentation, and interaction has been used loosely to describe a visualization technique or tool. Here, a formal background is provided. In addition, these concepts are later applied in the design of our prototype.

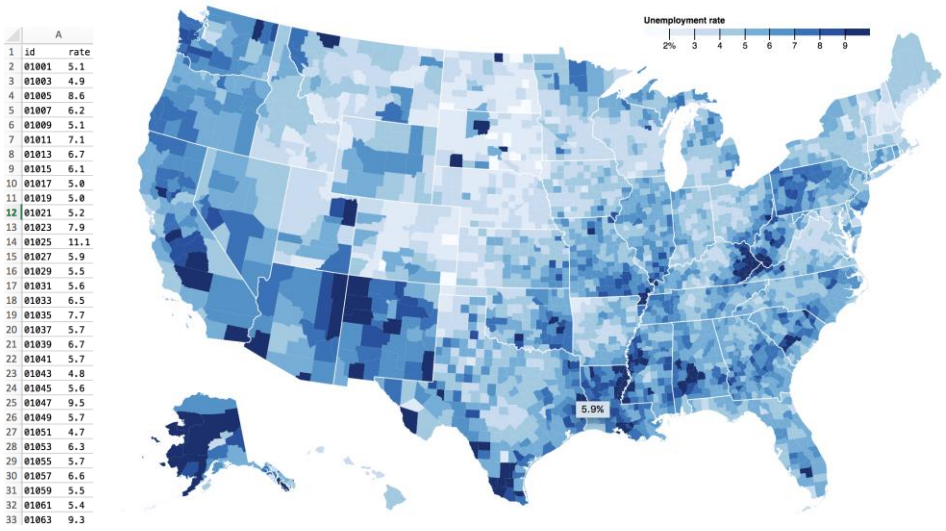
3.1 Information Visualization

The term 'visualization' is often used loosely to mean many concepts including visualization tools, visual representations, and the process of forming mental images. Spence (Robert Spence, 2014) defines 'visualization' closely related to its dictionary counterpart, that is, 'visualization' refers to the formation of mental images and is *internal* to human beings. However, in the literature, there is no definition that is commonly agreed upon (Parsons & Sedig, 2014). For instance, in (Card, Mackinlay, & Shneiderman, 1999), the term 'visualization' is defined as: "the use of computer-supported, interactive visual representations of data to amplify cognition". In this thesis, the term is used for its general definition including meanings for visual representations, and visualization tools, etc.

Information Visualization (InfoVis) is a field that is concerned with using visual representations of data to reinforce human cognition (R Spence, 2002). That is, we are concerned with gaining insight from data by means of visualization tools. As mentioned earlier, computers are very good at solving specific, or well-defined, problems. However, part of the goal of an information visualization is to assist the human in data *exploration* (Keim, 2002). Here, the problem is more ill-defined, and often the tasks involve activities like browsing, exploration, and gaining insight (Fekete et al., 2012). Often, the user doesn't know at hand the specific goals or problems, and it is through navigation and exploration of data, does insight emerge.

One of the main reasons to apply InfoVis is to take advantage of human perception. Through perception, a visual display provides the highest bandwidth of information from the computer to the human (Ware, 2012). This has strong implications in fields with very large amounts of data. For instance, in Figure 9, the left shows a textual representation of unemployment rate in excel,

and a visual map representation (choropleth map[17]) on the right. It is almost impossible to quickly see patterns or insights from the excel table, but the visual map instantly reveals areas of interest. Among others, a well-designed visualization has a number of benefits in dealing with data: they provide an ability to comprehend large amounts of data; they allow perception of emergent properties that were not anticipated; make data become immediately apparent; and facilitate in understanding both small-scale and large-scale features of the data (Ware, 2012).



	A	
1	id	rate
2	01001	5.1
3	01003	4.9
4	01005	8.6
5	01007	6.2
6	01009	5.1
7	01011	7.1
8	01013	6.7
9	01015	6.1
10	01017	5.0
11	01019	5.0
12	01021	5.2
13	01023	7.9
14	01025	11.1
15	01027	5.9
16	01029	5.5
17	01031	5.6
18	01033	6.5
19	01035	7.7
20	01037	5.7
21	01039	6.7
22	01041	5.7
23	01043	4.8
24	01045	5.6
25	01047	9.5
26	01049	5.7
27	01051	4.7
28	01053	6.3
29	01055	5.7
30	01057	6.6
31	01059	5.5
32	01061	5.4
33	01063	9.3

Figure 9 Employment rate in Excel (Left) and a Choropleth Map (Right)

The visualization process involves both the computer and the human. There are 4 main stages that form a feedback loop, namely: data collection, data preprocessing, visual display of data, and human visual processing. Figure 10 (Ware, 2012) shows the stages in a graphic form. In the context of ontology visualization, the data collection stage involves ontology engineering from domain experts to curate the ontology. This thesis is concerned with the other three stages. That is, we first preprocess the ontology with *derived data*, and transforming it into an appropriate format. Next, the processed data is passed through a graphic algorithm to generate the visual displays. Because this is an *interactive* system, users can interact and manipulate the generated displays to further

[17] http://bl.ocks.org/mbostock/4060606

transform the displays. This loop of incremental user interaction and visual display transformation forms the feedback loop that ultimately support the *exploration* of the ontology.

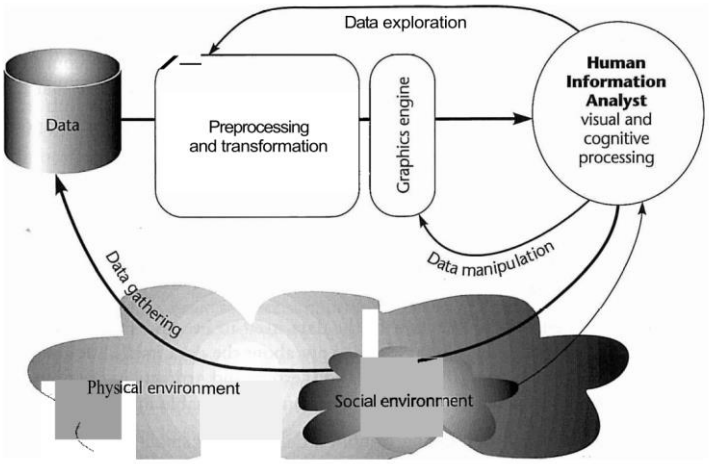


Figure 10 4 Stages of Visualization Process

3.2 Representations

The visual display of encoded data is the *visual representations (VR).* Representations can be internal or external (Scaife, Rogers, & SC, 1996). Internal representations refer to the human's mental model of a concept. External representations refer to representations of data through media, this can be pictorial graphs in a newspaper, or a VR on the computer screen. The internal representation of humans is often incomplete or contain errors (Tversky, 1993). Thus, it is through the interplay of external aid (e.g. VRs), and interactions (i.e. a feedback loop), does the mental model of the human become more complete and contain less errors. In our context, an ontology user may have little to no knowledge of the ontology's structure. That is, the user may have a weak mental model of structural information such as the ontology's depth, breadth, general size at each level, or the key terms at each level. It is our goal to design effective VRs and interactions to support the closing of the gap between the user's internal mental model and the underlying ontology.

The design of VRs is based on the idea of abstraction of some entity or concept. This abstraction is usually encoded visually in many forms such as position, length, or color hue. In the example of a line graph, we abstract a set of numerical data points onto a single line with varying positions.

This process of mapping data attributes into a visual form is called *encoding* (Robert Spence, 2014). The design of representations depends heavily on the data type. Shneiderman (Shneiderman, 1996) lists 7 datatypes, namely, 1-dimentional, 2-dimentional, 3-dimentional, multi-dimensional, temporal, tree, and network. To support the encoding multidimensional data, *visual variables* is used to encode each dimension. Figure 11 shows a set of visual variables. The choice of visual variables can have different influences on perception depending on the data type. For instance, using color hue to encode quantitative data may be less appropriate than compared to encoding ordinal data. To aid in the choice of selecting visual variables, Mackinlay (Card et al., 1999) outlined a ranking of most visually distinguishable visual variables for quantitative, ordinal, and nominal data types in Figure 12.

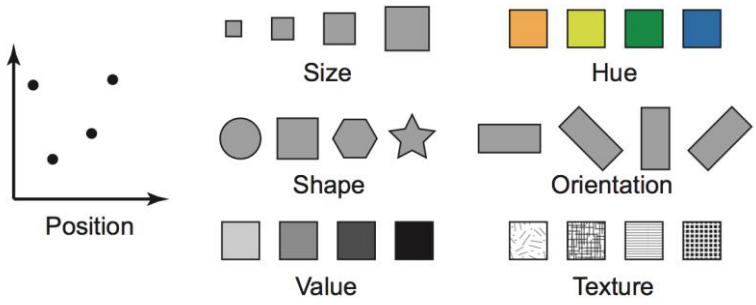


Figure 11Visual variables

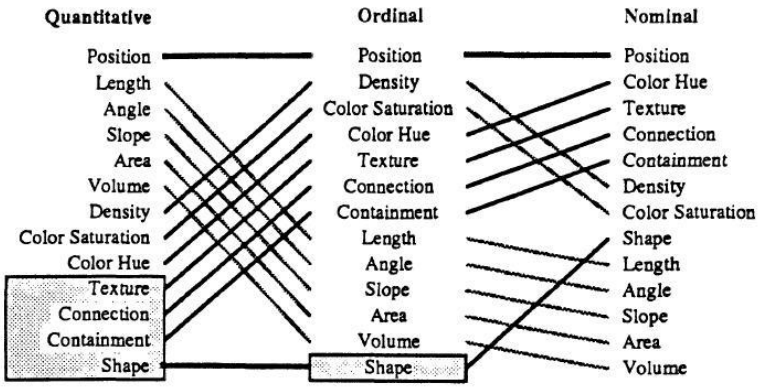


Figure 12 Appropriateness of visual variables based on data type, adapted from (Card et al., 1999)

3.2.1 Cognitive Influences of Visual Representations

The design of a new visualization system is often a creative process. There is not a perfect or definitive way to determine visual representations that will best suit the data. Generally, there is an element of *trial-and-error* in mapping data types and properties onto different visual encodings. For instance, in designing a visual representation of a multi-dimensional dataset containing a tree structure, which combinations of visual variables and representation techniques should be considered? Should we encode entities in a node-link layout or a treemap layout? Many designers may base their decisions on intuition, inspiration from past works, or simply by visual appeal. However, without a systematic design thinking, finding a suitable representation can be time consuming or prone to unsuitable designs. Moreover, only designing a beautiful visualization without considering its cognitive consequences may provide little to no benefit in supporting a user to *understand* the data (Hansen & Johnson, 2004). In addressing this issue, Sedig and Parsons (Parsons & Sedig, 2014) explored the perceptual and cognitive influences of common visual representation designs. In their works, a comprehensive and systematic framework of visual representations patterns and cognitive utility of common visualizations is developed to better support a visualization designer's choice of representations. Below is a summary of their works in 'Common Visualizations: Their Cognitive Utility (Parsons & Sedig, 2014)' and "Design of Visual Representations for Human-Information Interaction: A Pattern-Based Framework (Sedig & Parsons, 2015)."

The building blocks of encoding information is through *visual marks.* Visual marks include atomic visual entities such as dots, lines, colour, or simple shapes. To visually encode a multidimensional entity, visual marks can be composed to form multidimensional icons or *glyphs.* A technique that uses this idea is the Chernoff faces (Chernoff, 1973). In Figure 13, each feature of the face (e.g. length of nose, width of mouth, or slant of eyes) encodes a different property of the dataset. Through this encoding, a user can quickly discern entities that have specific values of a property. For instance, if the slant of the mouth encodes happiness rating of a neighborhood, it is easy to visually detect faces that are smiling. Indeed, *glyphs* exploit the perceptual system's ability to quickly discern spatial relationships and differences (Parsons & Sedig, 2014). More specifically,

according to the *Semiotics Theory*[18], stimuli are processed in two phases: *pre-attentive processing*, and, *post-attentive processing* (Eco, 1977). The *pre-attentive* stage refers to the near instantaneous (≈250 ms) perceived impulses such as shape or colour. And the *post-attentive* stage refers to the emergent features detected after conscious cognitive processing. Accordingly, glyphs can support both the quick detection of visual differences in addition to facilitating higher-order cognitive processes due to their emergent features (Parsons & Sedig, 2014).

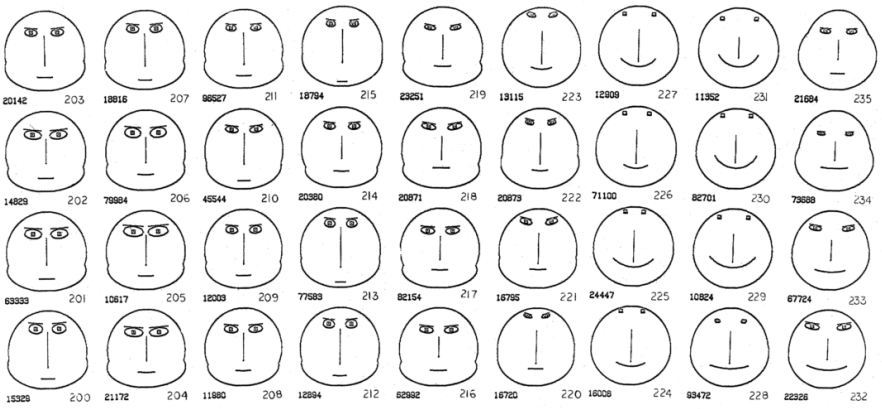


Figure 13 Chernoff Faces for Geological Data, adapted from (Chernoff, 1973)

Beyond glyphs, visual marks can be placed spatially based of different coordinate systems to show relationships among entities. Some examples of these spatial arrangements include plots and charts, maps, trees and graphs, and enclosure diagrams. In choosing a spatial arrangement visual marks, the underlying data structure must be examined (e.g. multidimensional, tree or graph); but also, and more importantly, the *cognitive* utility of these common visualizations should be reviewed. A plot or chart maps information onto a coordinate system. In the case of a scatterplot, this coordinate system is a 2-dimensional X and Y linear system. A plot can facilitate the perception of deviations and outliers, and consequently, this can support cognitive activities that involve reasoning about patterns or trends. A map is a coordinate system based on a geological map where visual marks are superimposed on top of. A map facilities high-level cognitive activities such as route planning or spatial navigation regarding geographic information. Trees or graphs spatially connect visual

[18] Semiotics refers to the study of signs and symbols and how they convey meaning (Eco, 1977)

entities by a line. The lines are readily detected by our perceptual system to reason about relationships. Finally, the enclosure diagrams place entities in different regions of space. An example of this is a treemap. Here, each region suggest commonality, and in the case of a treemap, all further subdivision under a subdivision belongs to a common parent. Cognitively, enclosure diagrams facilitate reasoning about set memberships or inclusion and exclusion of information (Parsons & Sedig, 2014).

3.2.2 Visual Representation Patterns

Hitherto, many visualization techniques have been mentioned, e.g. chernoff faces, scatter plots, treemaps, node-link graphs, and choropleth maps. Each of these techniques generally require a very specific data format or structure for the technique to apply. For instance, scatter plots require two numerical properties of the entity for each axis; choropleth maps require geological data coupled with another numerical density; and treemaps require a parent-child relation among the entities. However, much of the datasets in the real-world may not exactly fit these specific formats or structures. Moreover, these visualization techniques may not adequately support the cognitive tasks required by the users. In these cases, new and novel visual representations should be developed to accommodate the users' requirements.

In their book "Design of Visual Representations for Human-Information Interaction: A Pattern-Based Framework", Sedig and Parsons [35], attempt to move towards a *science* of visual representations. In doing this, a pattern-based framework is developed from unifying previously relevant ideas and reviewing thousands of visualization techniques. This pattern framework contains a set of 14 representations patterns in which specific techniques are *instantiated* from. Here, a pattern is not a directly usable pattern but rather an *abstract* idea of a whole class of representation techniques. As an example, one of the patterns is *hierarchy*. *Hierarchy* is characterized by: "[Mapping] information items onto VRs and organize them in a hierarchical, multi-level, pyramid-like fashion, where higher levels are superior to or contain and encompass lower level VRs [35]." This characterization contains techniques such as treemaps and node-link trees, as they both implement this *hierarchy* pattern. The framework becomes powerful when multiple patterns are blended to create new and novel visualizations. As an example, the scatter plot instantiates the *coordinate* and *token* pattern. Knowing this, we can theoretically further

extend our scatter plot by integrating the *link* pattern and connect entities via a line (should the data fit suitable). In result, these abstract patterns provide a comprehensive and systematic way to support the process of designing new and novel visual representations. The remaining patterns are detailed in Appendix A.

3.3 Presentation

After the design of visual representations, they need to be *presented* to the users. That is, *presentation* is concerned with how to spatially display the visual representations on a screen. For instance, if we decided to encode an ontology in the form of a node-link graph representation, how should we arrange the nodes and links to properly display the graph? In the case of large ontologies, this graph will clutter the screen no matter how we try to fit or arrange it. This *presentation problem*, i.e., trying to fit a very large amount of information on a very small screen (Robert Spence, 2007) is central to many presentation techniques. Because there is often no way to fit everything on the screen, unimportant or irrelevant information must be temporarily omitted. A simple example is *scrolling.* In a Word document or a mobile device, scrolling is used to shift information in and out of the screen and only present the relevant page of information at any instance. Other more complex presentation techniques often include *distortions* or incorporate *interactions.* A few examples of these techniques are presented below.

3.3.1 Focus+Context distortion

A problem with simply scrolling information out of the screen is that users may lose *context.* Here, the *context* refers to the set of information directly on the edge of our data of interest. In a node-link graph, the context may contain neighboring nodes that are a few depths away. Or in a zoomed in geographic map, the context may be the province or country in which the current city is focused. A technique to address this issue *is focus+context*, or *distortion.* The idea behind focus+context is to expand the area of interest (i.e. the focus), and at the same time, distort or minimize the surroundings (i.e. the context). An example of this technique is shown in The Perspective Wall (Mackinlay, Robertson, & Card, 1991), where a bifocal display is implemented to show focus+context. In Figure 14, the Perspective Wall displays a set of documents, files, phone calls and emails over a timeframe. The center is the focus, representing the current data of interest, while the sides are the distorted context showing past or upcoming items. The Perspective Wall only

distorts on the y-axis, but we can extend this idea and further distort the x-axis. This distortion of both the x and y-axis if often referred to as the *fisheye lens* effect (Gutwin & Fedak, 2004). In Figure 15, a Cartesian plane with straight lines is applied with *fisheye* distortion near the left center. Here, only the cell under the 'lens' is focused while the rest is more and more distorted smaller.

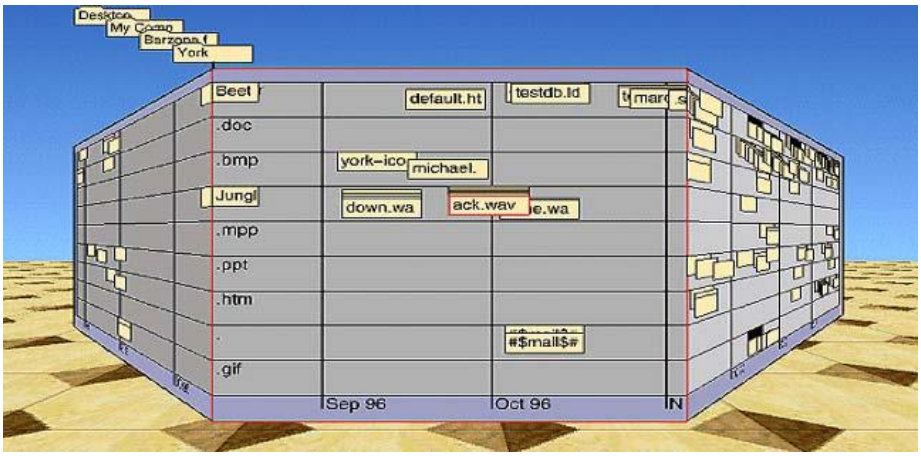


Figure 14 The Perspective Wall with focus+context distortion

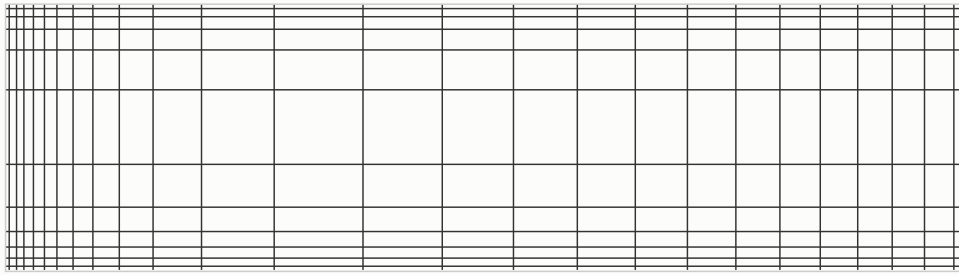


Figure 15 Fisheye distortion both the X and Y axis

3.3.2 Magic Lenses

The fisheye lenses are part of a realm of techniques known as *magic lenses.* The *magic lenses* are based on a metaphor of placing a physical lens (either circular, square, or other shapes) on top of a representation, and the items directly under the lens would be distorted in some way—much like a magnifying glass. However, *magic Lenses* can do more than just spatial distortion. In Tominski et al.'s survey (C Tominski, Gladisch, Kister, Dachselt, & Schumann, 2016), some other applications of *magic lenses* include *suppression* and *enrichment*, in addition to *alteration*. In a *suppression* lens, the data under the lens is omitted, or *filtered* based on a set of queries. This

suppression of extra information can help clear up clutter and reduce occlusion (Figure 16, left**)**. An *enrichment* lens, on the other hand, provides extra information for the items under then lens. For instance, in Figure 16 (right), text labels are only shown for nodes under the lens, as we cannot show the text labels for every node by default.

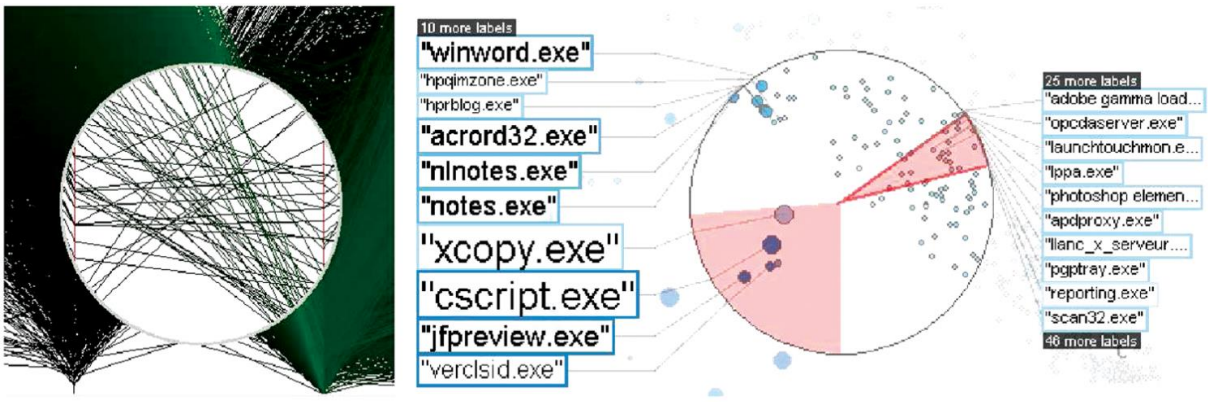


Figure 16 Left: Suppression lens, Right: Enrichment lens, adapted from (C Tominski et al., 2016)

3.4 Interaction

The *magic lenses* are not static. That is, the lenses can be dynamically moved across visual representations by the user, and the computer will update the representations accordingly. This dynamic interplay between the computer and user is studied in the field of *Human-Computer Interaction (HCI).* Although HCI draws research from many fields—including psychology, cognitive science, visual analytics, and interaction design (Dix, Finlay, Abowd, & Beale, 2004)—in this section, we focus on the interactions on visualizations, and how they support cognitive tasks.

As previously mentioned, to avoid *information overload*, only a subset, or a specific view-point, of the complex dataset is presented at any one point. However, this limited view is often not enough to convey every aspect of large or complex datasets. To help mediate this problem, *interactions* is used to 'construct' and 'reconstruct' new graphics until all relationships constituted by the data is revealed (Bertin, 1981). The role of *interactions* is therefore heavily coupled with the human to support the *exploration* of data (Christian Tominski, 2015). That is, there is an ongoing loop of actions and feedbacks (Christian Tominski, 2015).

In HCI literature, this action-feedback loop is characterized by the *gulf of execution* and the *gulf of evaluation*, first coined by Norman in (Norman, 1988). The *gulf of execution* refers to the gap between what the user perceives are possible actions and what the system is possible of providing. And the *gulf of evaluation* refers to the gap between what the system provides as a feedback (in our context, in the form of new or updated visual representations) and what the user's expected results are (Norman, 1988). This model can be further extended with the execution phase starting with a *goal*, then an *intent* to interact, and then a mental planning of interaction, and finally performing the action (Christian Tominski, 2015). Further, the evaluation phase starts with a response from the system, and then the perception of the response, and then the interpretation of the response, and finally the evaluation of the response. Figure 17 provides a schematic representation of the twin gulfs. It is the goal of the interaction designer to minimize, or *bridge*, the twin gulfs of execution and evaluation to allow a user to *efficiently* and *effectively* (Christian Tominski, 2015) carry out their visualization tasks.

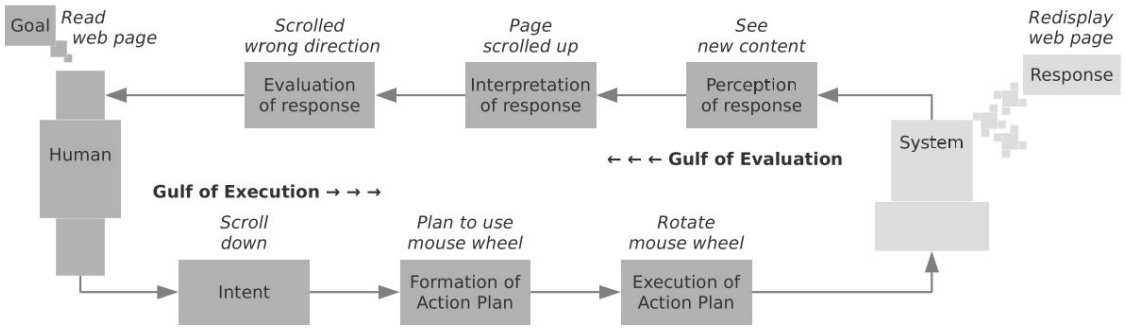


Figure 17 Interaction as gulf of execution and evaluation, adapted from (Christian Tominski, 2015)

3.4.1 Levels of Interaction

The gulfs of execution and evaluation presents a very high level idea of how interactions should work, i.e. an action from the user and a reaction (feedback) from the system. However, this description is not enough to help communicate the large diversity of possible interactions. For instance, consider the interaction of *comparing*, do we mean *compare* at a low level of perceptually comparing two visual representations, or *compare* at a high level as a cognitive task resulting from multiple interactions? A step forward is to classify, or characterize the interactions based on levels of abstraction. Ware (Ware, 2004) describes the interaction loop in three levels: low-level,

intermediate-level, and high-level interaction (Christian Tominski, 2015). Low-level interaction (or *data manipulation loop)* refers to primitive actions such as moving or clicking the mouse. Intermediate-level interaction (or the *exploration and navigation loop)* refers to the combination of low-level interactions to *explore* and *navigate* a large visual data space. And the high-level interaction refers to processes such as problem-solving or falsifying hypothesis about the data (Christian Tominski, 2015).

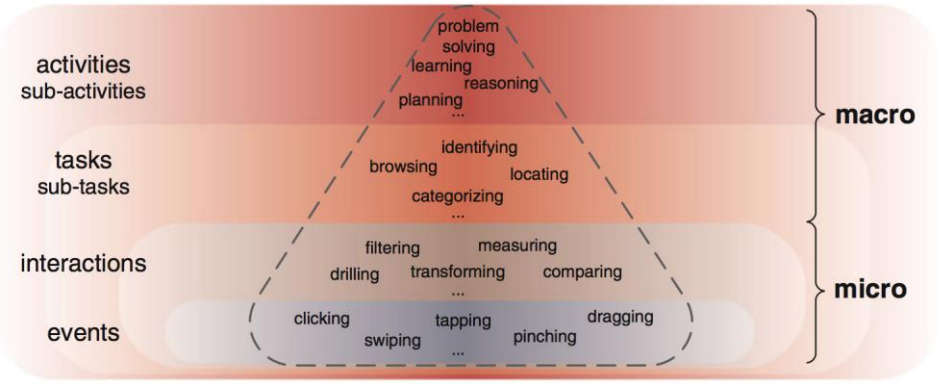


Figure 18 Categorization of Interaction, adapted from (Sedig, Parsons, Dittmer, & Haworth, 2013)

Sedig et al. further extends this classification of interactions into four main levels: *events, actions, tasks, and activities* (Sedig et al., 2013) (see figure 18). Here, *events* refer to the building blocks of interaction, e.g. mouse clicks and keyboard presses; *actions* refer to the what is performed upon an interface, e.g. filtering, and drilling; *tasks* are activities that are goal-oriented and often require a combination of actions, e.g. browsing, and locating. Finally, activities are at the highest level that are often complex and open-ended (Sedig et al., 2013). Conceptually, higher level tasks and activities only emerge from a combination of repeated lower level interactions.

3.4.2 Interaction Patterns

Of the four levels of interactions described earlier, we are more concerned with the *actions (or interactions)* level. Because it is at this level that we can design actions and reactions in which the users can directly act upon. Some interactions mentioned earlier include filtering and drilling. As an example, we can implement drilling in a geographic map visualization by means of mouse wheel scroll to zoom in and out the map. But are there more interactions? And what about their cognitive utilities? Much like the pattern framework of visual representations, Sedig and Parsons

(Sedig & Parsons, 2013) approach the design of interactions systematically through a pattern based framework called EDIFICE-AP[19].

Here, we make a distinction between *pragmatic* and *epistemic* actions. *Pragmatic* actions set out to achieve a physical goal, e.g. pushing open a door. However, *epistemic* actions aim to transform the world to facilitate mental processing (Kirsh & Maglio, 1994; Sedig & Parsons, 2013), e.g., zooming (or drilling) into a visual interface to bring out latent information to support cognitive tasks. The framework EDIFICE-AP consists of 32 *epistemic* action and reaction patterns that aim to facilitate the systematic design of interactions. These actions are general enough to be independent of specific implementation styles, yet specific enough to stimulate the systematic thinking of interaction design for complex cognitive activities. Here, we present the pattern of *Collapsing/ Expanding* as an example to show its application. The remainder patterns are detailed in Appendix B.

The *Collapsing/ Expanding* pair is characterized by the folding an unfolding of visual representations. This is often useful for representations that are very dense in information. Indeed, by reducing the complexity of a visualization, it also reduces the perceptual and cognitive burden it places on a user (Sedig & Parsons, 2013). An example of this pattern is used in the SpaceTree (Plaisant, Grosjean, & Bederson, 2002), a visualization to support the exploration of large node-link trees. In the SpaceTree, subtrees can be subsequently collapsed and expanded to and from an icon that encodes the depth and breadth of the subtree. By collapsing subtrees that are unimportant to the user, the visualization reduces clutter and occlusion on the screen and in turn better aid the exploration of tree data structures.

3.4.3 Interactivity

As mentioned earlier, the interaction patterns are characterized at an abstract level independent of implementations. That is, each of the interaction patterns can be *operationalized* through various

[19] Epistemology and Design of human-InFormation Interaction in complex Cognitive activitiEs (EDIFICE) – Action Patterns (AP)

techniques (Sedig, Parsons, & Babanski, 2012). For instance, the *arranging* pattern (the reordering of representations spatially or temporally) can be operationalized by ways of keyboard clicks, using mouse clicks on buttons, or directly dragging on the visual representations via a mouse. This discrepancy in the way a pattern is operationalized can either help or hinder in the coupling of a user's internal representation with the external visual representations (Sedig et al., 2013). For example, if an arranging action is performed but the feedback is slow, sluggish, or non-existent, then this would directly lead to poor user experience. A step towards designing effective interactions is to recognize the *quality* of interactions—also referred to as *interactivity* (Parsons, Sedig, Didandeh, & Khosravi, 2015).

In characterizing *interactivity*, it can be classified into two levels, i.e. at the granularity of *micro* and *macro* levels (see Figure 18). At the *micro* level, it is concerned with the interactivity of lower-level interactions including the action-reaction patterns described in the EDIFICE-AP framework. And at the *macro* level, it is concerned with interactions at the higher levels, including tasks and activities. In other words, *macro* level interactivity is concerned with the overall quality of interaction from the combined whole of the lower level interactions. To aid in the characterization of interactivity in both the micro and macro levels, Sedig et al. (Sedig et al., 2012) developed a conceptual framework called EDIFICE-IVT[20] with 12 characterizations for the actions and reactions (micro level) and 5 characterizations for macro level interactivity. Here, we describe the interactivity element of *granularity* for actions, and *activation* for reactions at the micro level, the remainder elements are detailed in Appendix C and (Sedig et al., 2013).

Granularity is concerned with the required steps an action needs to fully activate itself. It can be either atomic or composite (Sedig et al., 2013). Atomic granularity requires only one step, whereas composite granularity requires more than one steps. As an example, to implement the drilling pattern on a geographic map visualization, we can implement it in two ways: 1) using mouse wheel scroll, and 2) allowing the user to specify the magnification size followed by a 'go' button. Here, the first method, a single wheel scroll, has an atomic granularity, whereas method 2 is composite.

[20] IVT stands for interactivity

Activation is concerned with the point at which a reaction begins after an action. It can be immediate, delayed, or on-demand. Immediate activation happens when the system reacts instantaneously to an action. Delayed activation happens with a temporal gap after the action. And on-demand activation only happens after a subsequent request by the user (Sedig et al., 2013).

In designing with interactivity considerations, there is often not one form of an element (e.g. atomic or composite granularity) that is better than the other. For instance, it is not always best to design with immediate activation, there are some cases in which delayed or on-demand activation can be helpful (Sedig et al., 2013).

Chapter 4 Prototype Design

In this chapter, we detail the design of an interactive visualization prototype for the exploration and navigation of ontologies. This prototype's general approach is to apply multiple coordinated views to support the ontology exploration process—hence the name MOE (Multi-View Ontology Explorer). Here, we apply design considerations into the prototype from previous chapters including the conceptual frameworks for visual representations, interactions, and interactivity, i.e., EDIFICE-AP, and EDIFICE-IVT, respectively. Finally, an overview of the implementation for MOE is provided.

4.1 Methods

In Chapter 2, we outlined the various tree and graph based visualization techniques used to visualize ontologies. Namely, these techniques include the indented list, node-link graphs, zoomable interfaces, space-filling, focus+context, and 3D interfaces. However, these techniques have their own individual shortcomings when an extremely large ontology dataset is applied. Most notably, these visualizations suffer from clutter and occlusion when they attempt to visualize a large portion of the ontology—or the information overload problem. Moreover, many visualizations suffer from the loss of overview, or orientation, when zoomed in or traversed. Many interaction and distortion techniques have been studied to mitigate the issue of information overload, some of these techniques include hyperbolic distortion, magic lenses, and node summarizing techniques (KC-Viz). Although these distortion and interaction techniques help mitigate information overload at a smaller scale, it still cannot fully scale to the full size of ontologies (tens of thousands of nodes).

In our prototype design, we take a multi-view (or separated views) approach to aid in the exploration and navigation of ontologies. Separated views have the advantage of combining different visualization techniques to overcome the drawbacks of visualizations in a single view. However, an issue with using separated views is to coordinate all the separate visualizations when one is updated (i.e. being interacted with by the user). A technique to address this issue is to apply

dynamic brushing and linking (Keim, 2002). That is, changes in one visualization is automatically and instantaneously reflected upon the other visualizations.

At each separated view, we visualize a different level of abstraction in the ontology from a high-level overview to a low-level entity. More specifically, we visualize five levels of abstraction which are divided with the following concerns: show an overview structure of the ontology; show entity and hierarchical parent relationships at an individual level; show shortest distances between landmark entities at a level; show immediate parent and children entities; and show the paths to reach an entity. Collectively, we name these visualizations V1 to V5, where V1 refers to the overview visualization, V2 entities at each level, and so on. In addition, we provide user interfaces to view the full details of an entity, and enable searching and saving (see Figure 19 V6, V7, respectively). An overview of our prototype is depicted in Figure 19.

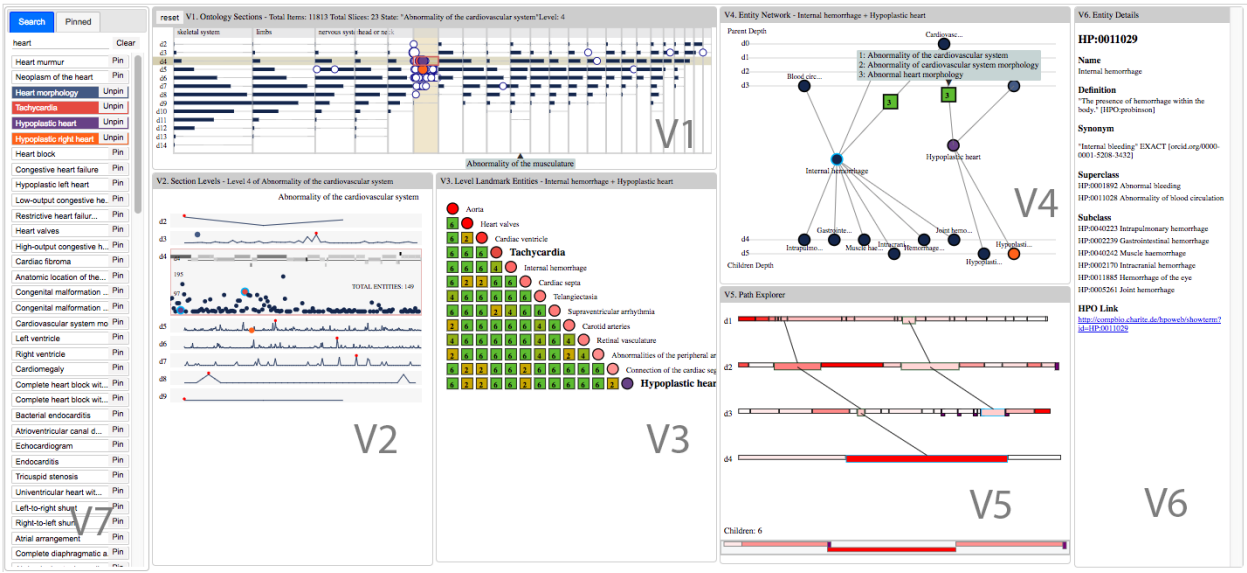


Figure 19 MOE overall interface. V1) overall structure, map V2) Nodes and Parents at each level V3) Shortest distance among landmarks in each level, V4) Immediate children and parents of a node, V5) All Paths to a node V6) full details and linking panel, and V7) search and pin interface

Lastly, to support the formation of mental models, i.e. the sense-making process, we employ a metaphor consistent with the formation of spatial mental models. That is, we encode specific entities in the ontology to support the formation of landmark, route, and survey knowledge. Each level of abstraction has a sense of landmarks relative to that abstraction. That is, at each level in

the ontology, we encode landmarks only relative to that level. In this prototype, we identify landmarks by their connectivity, or the total number of accessible nodes. In graph theory, this measure of importance is the transitive closure of a node. Indeed, other measures of importance can be used in place of the landmark value, e.g. betweenness or closeness measures (White & Borgatti, 1994).

The general flow of exploration in MOE to support the process of sense-making is as follows. First, the user can search for a specific term (Figure 19, V7) and explore the results' locations, children nodes, parent nodes, and surrounding entities, among others. However, often novice users don't any have previous knowledge of the ontology, and in this case, the user may start the exploration process in V1. In V1, we provide a general overview of the ontology's structure in terms of total major subsections (henceforth also referred to as a slice), depth (i.e. levels), breadth, and nodes at each level. A user can select a slice's level and explore the nodes and landmarks under that level in V2. Alongside V2 is V3 that provides the top landmarks at the level selected in V2. In V3, we encode for the shortest paths among these landmarks. The purpose of V3 is to quickly show landmarks that are very close together or far away. Digging deeper, a user can select a specific landmark in V3 to activate V4 and V5. In V4 we show the surrounding nodes of one or two landmarks that are above it (i.e. immediate parents), and nodes directly below it (i.e. immediate children). And in V5, we focus on a single node to show all the specific paths and parents to reach that node. Finally, we list all the detail properties of an ontology entity in V6.

Outside this exploration loop, we provide interfaces to search and pin any items in the ontology. That is, a user may explore one path in the ontology and choose a pin one or more nodes, and start the exploration process down other paths, and further pinning more nodes. These pinned nodes persist throughout all the visualizations in a colour coded manner. This way, a user can always see their items of interest fit inside the ontology, and how they compare to other key landmarks.

In this thesis, we demonstrate our prototype with the Human Phenotype Ontology[21] (HPO). In HPO, each term describes a phenotypic abnormality such as skeletal system abnormalities, nervous

[21] http://human-phenotype-ontology.github.io/

system abnormalities, or abnormalities of the eye. For the remainder sections of this chapter, we provide details for each visualization from the context of HPO.

4.2 V1: Ontology Overview

In the first level of abstraction, V1 provides an overview of the basic structure of the entire ontology. These structural components include the total number of major subcomponents (or subtrees) under the root, the depth, and the breadth on the ontology. These major subcomponents provide a sense of the most general terms or key landmarks at the lowest distance from the root. The depth component is concerned with which subtree or path is the deepest or shallowest from the root. And finally, the breadth is concerned with how many items are at each level for each subtree. These structural components can give a very quick sense of the general shape of the ontology, e.g. if an ontology is very deep but do not have a lot of breadth, or conversely, a very shallow ontology but a large breadth at a specific level.

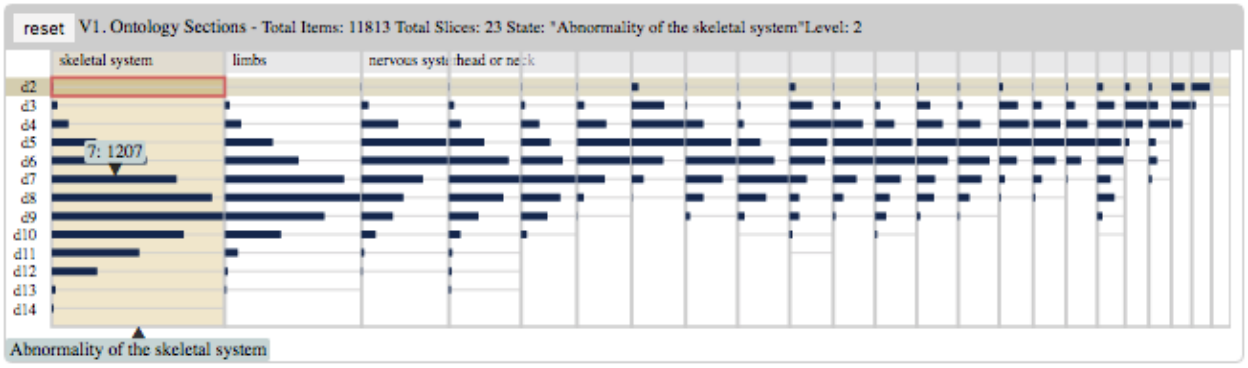


Figure 20. V1 Ontology Overview. Visualization of overall structure.

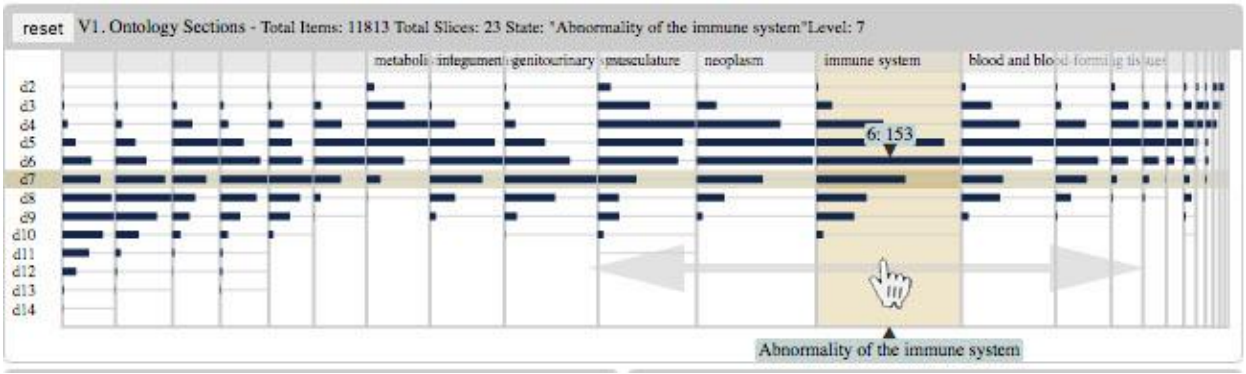


Figure 21. V1 activated with distortion lens. The slice ‘immune systems’ is under focus.

Figure 20 depicts V1 at initialization. In its initial state, we subdivide and sort each section relative to the size of each subtree from the root. This subdivision is analogous to a treemap subdivision of nodes. That is, the subtree with the most number of nodes is encoded with the largest width for its *slice*. Henceforth, slice, major subtree/subsection, may be used interchangeably to refer to V1's representation of major subsections. In HPO's V1 representation, the largest slices include abnormalities for the skeletal system[22], limbs, and the nervous system (Figure 20, top 3 slices on the left).

Depth from the root is encoded as horizontal lines starting with depth 2 to the deepest level. Depth 1 is omitted here because they are itself the major slices. Here, we can see *d14* (left axis), or a depth of 14, is the deepest path belonging to the slice of skeletal systems. Scanning quickly across V1, we can get a sense of the relative depths of other slices, i.e. most slices have a depth of around 9 or 10.

Within each slice is a histogram style representation (in blue bars) that encodes the relative number of nodes at each depth. This representation aims to give an overview of the level with the most or least nodes. The length of each blue bar encodes the number of nodes. The bar stretching the entire width of the slice is the level with most nodes. For instance, under skeletal systems, level 9 has the most nodes, followed by level 8. If a user wish to see the exact number count for each level, a tooltip is provided on mouse-over interaction. From Figure 20, hovering over depth 7 under skeletal system tells us that there are 1207 nodes at depth 7.

An issue with V1 in its static form is the occlusion of very small slices. These slices occur when the largest slice and the smallest slices have a large discrepancy in node count. In the case of HPO, the largest slice 'skeletal systems' have over 3000 nodes while the smallest slice 'thoracic activity' has less than 10 nodes. Due to this discrepancy, the widths of smaller slices can become very small; i.e. if scaled linearly, these small slices can get widths with less than 1 pixel on screen. To address

---

[22] In this V1, the slices are labeled with names such as 'skeletal system' and 'limbs'. However, the full name in the HPO for these terms are 'Abnormality of Skeletal Systems' and 'Abnormality of the limbs'. We remove the redundant leading terms 'abnormality of' for brevity and clarity.

this issue, we first scale the subdivisions linearly with a minimal width mapping to the slice with the least nodes. With a minimal width, we can visually discern the different slices even if they are considerably smaller than the largest slices.

Another issue in the static form of V1 is the lack of details for the smaller slices. Although we can discern the presence of these smaller slices, it is hard to see any further details. For instance, in figure 20, the slices after the third one are not big enough to show their names. To address this issue, we use a distortion magic-lens to allow a dynamic focus of a slice while distorting the surrounding slices, and thus keeping context. To activate this lens, the user must press and hold down the shift-key while hovering over a slice of interest. As a user moves the cursor, the slice directly under it is distorted larger into focus while minimizing the surrounding slices. In figure 21, we apply this lens to see the smaller slices 'immune system' and 'neoplasm' in more detail. To deactivate this distortion, we provide a 'reset' button to restore the original subdivision sizes. Lastly, a tooltip with the full name of the slice is shown on mouse-over of a slice (see figure 21).

4.2 V2: Subsection Levels

In the second level of abstraction, we narrow down to a specific subsection, or slice. Here, we are concerned with the overview of nodes residing at each level within a slice. This overview includes the key landmark at each level, each node's landmark value relative to other nodes in the level, and the parents that contribute to the nodes at that level. In addition, we encode the distribution of important nodes across each level.

There are two methods to reach V2 from V1. A user can select an entire slice, or a specific level within a slice. If a slice is selected in V1, V2 is initialized to a collapsed form (Figure 22, left), otherwise, if a level is selected in V1, that level is activated in the expanded form (Figure 22, right). Regardless, users can expand or collapse levels thereafter by toggling the selected level in V1 on or off.

V2's visual representation is designed to closely resemble V1. That is, in V2, each level, labeled *d2* to the last level, *d14*, matches closely with the level blue bars in V1. This consistency aims to support a quick transition from V1 to V2 and vice versa.

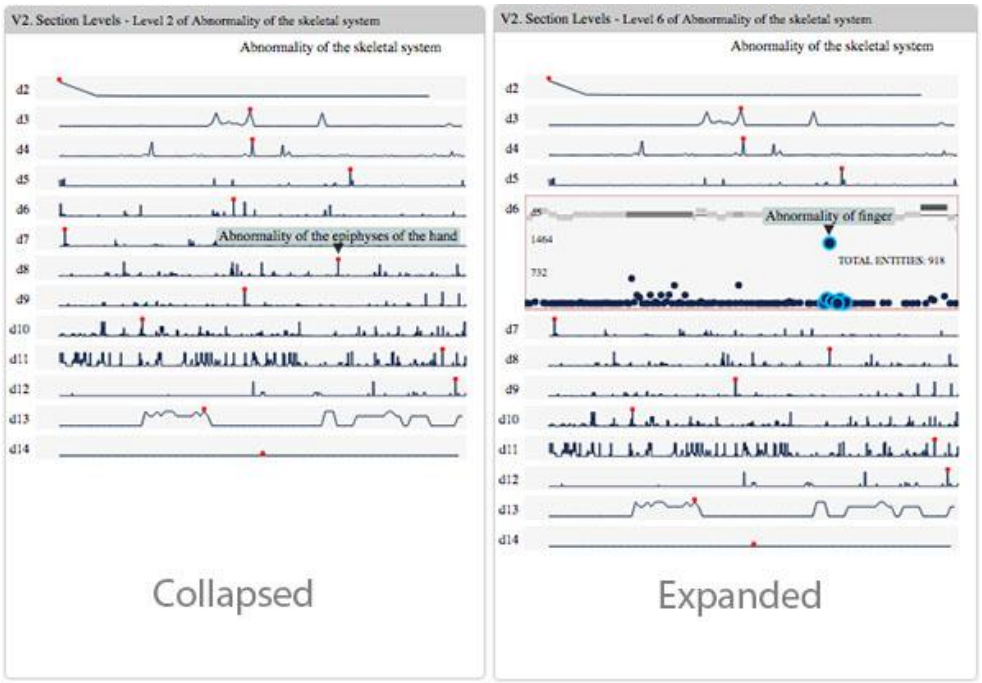


Figure 22 V2. Overview of nodes for each level with a slice. Left: all levels collapsed. Right: Level 6 is expanded.

Figure 23 V2. Distribution of landmarks

In the collapsed form of V2, it presents an overview of landmark distributions across all levels. In Figure 22 (left), we can see a collapsed V2 for the largest slice in HPO 'Abnormality of Skeletal Systems'. Here, we can see 14 levels with their subsequent landmark distributions in the form of a line graph. In this line graph, we encode the top landmark for each level as a red circle, with a tooltip to show this landmark's full name on demand. This line graph is mapped as follows. The y-axis encodes for the landmark value, i.e. total accessible children. And the x-axis encodes the individual position of each node grouped by parentage, i.e., nodes that are from the same parents are placed next to each other in a group. In this collapsed form, we can quickly see the top landmarks at each level, as well as the relative distribution of landmarks. For instance, in Figure 23, we see two levels within a slice, one at depth 7 and other at depth 8. For depth 7, we have a sense that there is only one significant landmark near the left (judged by perceiving a single spike).

However, in level 8, there are multiple landmarks, i.e. we see spikes in the left, a few in the middle, and one near the end.

V2 provides the exploration of a level by expanding a specific level panel (see Figure 22 right). Here, users can select from level to level, expanding and collapsing. The expanded form of a level is depicted in Figure 22 (right). In the expanded form of a level, we show more details of the nodes at that level. These details include the approximate landmark values the nodes have, the parents that contribute to the nodes at this level, and the approximate depths of these parents.

In Figure 24, we show the 'Skeletal System's slice expanded at level 3. Here, we see there are 63 entities (nodes) at that level. These 63 nodes take form in dark blue circle. The circles position vertically based on their landmark value. For this value, we draw 2 lines to approximate the average and max range, i.e. the 48, and 96 line. For instance, from this scatter-plot like graph (Figure 24, bottom), we can see there are roughly 2 landmarks near the center with around 96 total children, while the rest of the nodes have well less than 30 total children.

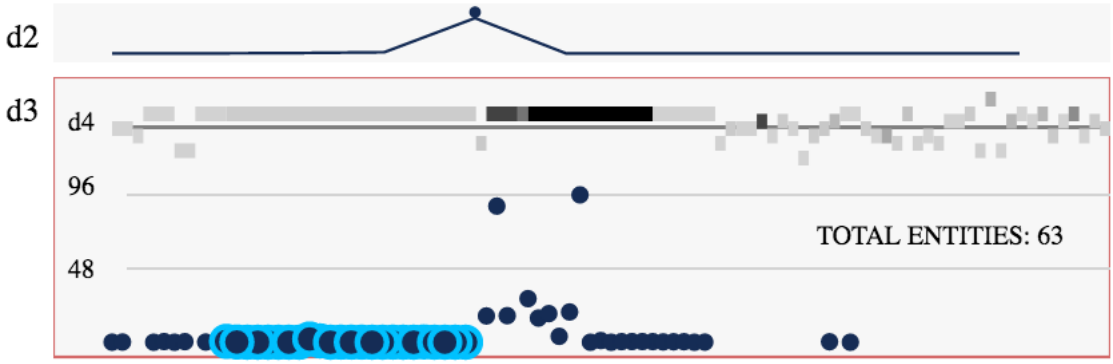


Figure 24 Expanded level 3 in detail. Circles encode nodes at level 3. Horizontal bars encode parents that contribute to the nodes directly under it.

Because V2 abstracts at the level of depth for the entire subtree, the nodes presented here can come from many parents. To encode the parents that contribute to nodes at this level, we encode them as horizontal bars (boxes) directly above the nodes. Each bar's width encodes total immediate children, which corresponds directly proportional to the nodes under it, i.e., all the nodes directly under it are its immediate children. And conversely, the parent of a node (circle) is directly above it. To better see this, when a user hover's over a parent, all the children are highlighted in blue (see

Figure 24). Because the ontology is a graph, these parents may come from many levels, and similarly, each node can have many levels due to multiple paths. Therefore, to encode for the approximate depth of these parents, we place the parent bar vertically based on their average depth. That is, the lower depth parents are near the top, and higher depth parents are near the bottom. The average depth is shown with a line. In Figure 24, we see that the average depth of parents has a depth of 4. Finally, the hue of the parent bars encodes the total children, where the darker colour encodes a higher count, and a lighter-grey colour encodes a lower count.

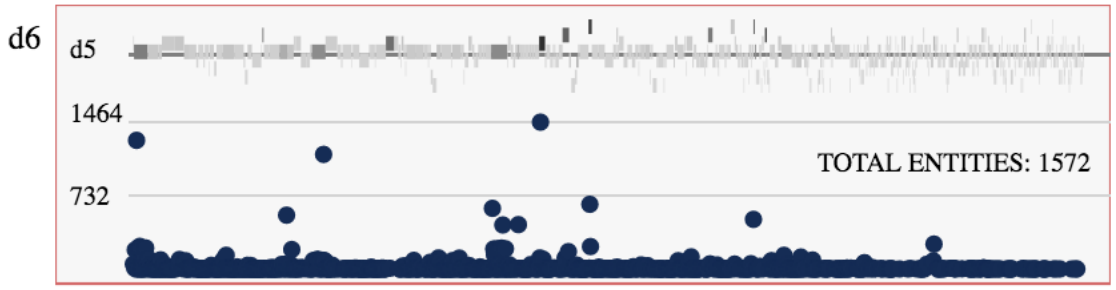


Figure 25 V2. Level 6 of skeletal system expanded. A lot of clutter and occlusion.

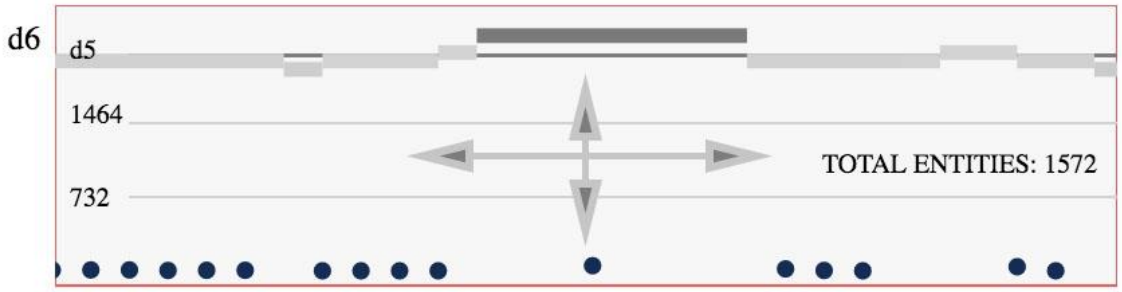


Figure 26 V2. Zoomed in level. Users and zoom and drag around the level.

An issue with V2 is that as the level get deeper, the number of nodes grow exponentially larger, and thus the expanded form would easily overflow with information. To illustrate this issue, we expanded the 6th level of 'Skeletal Systems' as depicted in Figure 25. In this level, there are over 1500 nodes, and consequently, also a high number of parents that contribute to these nodes. Because of the high number of visual representations, we can only discern the visually distinct items, i.e. the y-position of the nodes, and the parents with the darkest hue. To mitigate this issue, we use the interaction pattern of drilling, or scaler zooming, to stretch out the visual representation horizontally. This interaction is implemented as scrolling (for zooming) and dragging (to move

around the zoomed space. Figure 26 shows a zoomed in level 6 of the ‘Skeletal Systems’. Here, the nodes become stretched outwards such that only the nodes of focus stay in the middle. Overall, V2 provides an overview of the nodes, landmarks, and parents at a specific level of a major subsection (slice) in the ontology.

4.3 V3: Landmark distances

V2 shows an overview of nodes and parents at a specific level, however, it does not show much information about nodes among themselves. For instance, in Figure 25, we can see 3 landmarks with around 1400 total children at level 6, all from different parents. However, from V2, it is difficult to see the route information among these landmarks. That is, are the landmarks far apart or close together? To address this issue, we design V3 as a companion visualization to V2 that encodes the distances among the top 13 landmarks for any level. V3 updates synchronously with V2, i.e., when a level in V2 expands, V3 updates to reflect that expanded level.

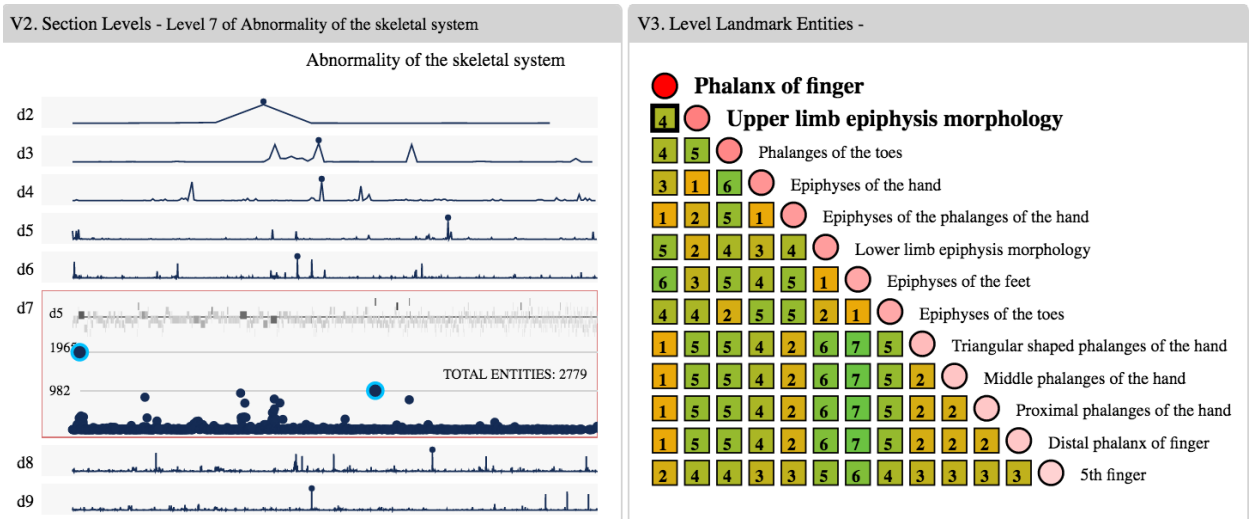


Figure 27 V3 (right) alongside V2 (left). Each cell encodes a landmark pair with their distances as a number. Distance is also encoded as a colour (orange for low, green for high distance). Selected cells highlight their name in bold, and highlight their position in V2 in blue.

Figure 27 (right) depicts V3 as a half matrix. Each row corresponds to a landmark; each cell corresponds to the distance between two intersecting landmarks; and each landmark is encoded as a circle diagonally on the matrix (i.e. intersecting with itself). Landmarks (circles) are sorted from top to bottom in the order of decreasing total children. This landmark value (total children) is also

encoded as hue, with red for the highest, and white for the lowest. For instance, from Figure 27, the top landmarks in level 7 of ‘Skeletal Systems’ include ‘Phalanx of finger’, ‘Upper limb epiphysis morphology’, and ‘Phalanges of the toes’, among others. Among these landmarks, each intersecting cell encodes the distance both as a number and with hue. For instance, in Figure 27, the top 2 landmarks have 4 nodes in the shortest path, while landmarks ‘Upper limb epiphysis morphology’ and ‘Epiphyses of the hand’ have 1. Distances of 1 and 2 show special relationships among the landmarks. A distance of 1 indicates that one landmark is the parent of another, while a distance of 2 indicate that they both share a common parent.

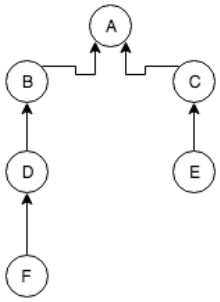


Figure 28 Simple hierarchy to demonstration distance algorithm in V3. Here, nodes ‘F’ and ‘E’ have a distance of 5.

To better illustrate this distance concept, Figure 28 shows a simple hierarchy with 6 nodes with the arrows showing parentage. In this example, nodes ‘F’ and ‘E’ have a distance of 5; that is, 3 from F to A [(F, D), (D, B), and (B, A)], and 2 from E to A [(E,C), and (C,A)]. Hence, this distance represents the number of nodes to pass in traversing the path from one node to another. In a realistic ontology, such as HPO, a node usually contains multiple paths (due to multiple inheritance), and so, in our design, we only encode the shortest path.

V3 is designed to be coupled with the expanded level of V2. The top 13 landmarks in V3 corresponds directly with the top 13 nodes from V2. More specifically, during dynamic linking, highlighted cells in V3 automatically highlights the 2 landmarks in V2. This idea is shown in Figure 27, where a user hovers the cell of the top 2 landmarks, and in effect, these 2 nodes are also highlighted in V2 in blue. This dynamic linking allows a user to quickly explore and see interesting landmarks that are close together or far away. In finding a landmark of interest, they can jump between V2 and V3 to view different levels of details.

4.4 V4: Entity Network

In V4, we reach a low level of abstraction, i.e., we are only concerned with just one for two landmarks in the ontology. The focus of V4 is to show the surrounding nodes of landmarks selected from V3. With this, users can explore the nodes around a set landmarks by traversing up or down the hierarchy through immediate parents and children relationships. In Figure 29. V4 is depicted with two landmarks ‘Tachycardia’ and ‘Hypoplastic heart’. The landmarks of interest are placed in the middle, and the parents and children are separated on the top and bottom, respectively. This separation of parents and children nodes are also placed on a y-axis scale to differentiate their average depths.

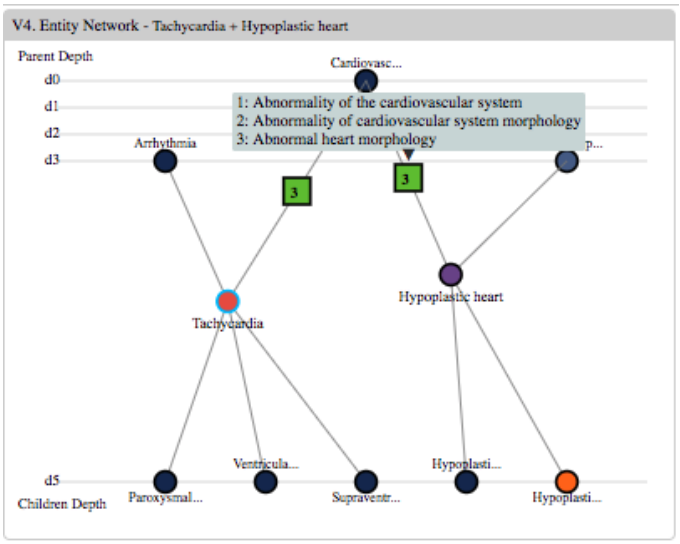


Figure 29 V4. Entity Network of two landmarks

V4 is dynamically linked with V3. More specifically, each update in V4 corresponds to a selection of a cell in V3. Because a cell in V3 only shows the distance among two landmarks, V4 provides the exact path to reach from one landmark to the other. In Figure 29, the cell of these two landmarks would simply state ‘6’ in V3. However, here, we can precisely see which path is longer, and what the exact nodes are in this path. More importantly, we can explore the common parent among these two landmarks, and the nodes that surround them. Finally, to see the full names of a node, a tooltip is provided on mouse hover.

4.5 V5: Path Explorer

The last visualization presented in the exploration process is V5. In V5, we are concerned with showing the paths to reach a single node. In previous abstractions (V1 to V4), they are generally centered around multiple nodes, parents, and their relationships. However, here, the focus is only on a single node. Due to this specificity, V5 only updates on the selection of a single node. In other words, V5 will only update from the selection of a node from V2's expanded level, V3's landmark circle, or any of V4's nodes. In Figure 20, we show the paths of 'Abnormality of the pulmonary artery'.

In V5, each level is separated vertically, with the lower depth at the top. Each level contains nodes with paths equal to that depth. Due to this mapping, nodes may be encoded in multiple levels (due to multiple paths). The node of interest is highlighted in blue; as shown in Figure 30, 'Abnormality of the pulmonary artery' appears both in level 4 and 5, with a total of 2 paths.

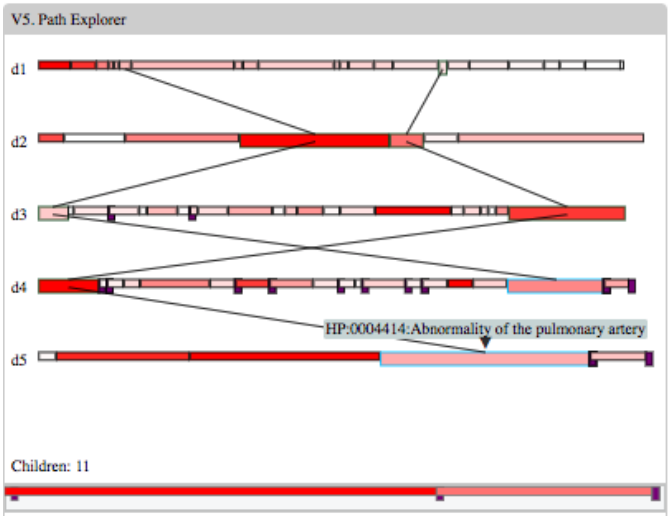


Figure 30 V5. Path Explorer shows all paths to a node. Width of a box encodes immediate children count. Hue encodes total children, with red being highest value at the level, and white being lowest. Purple protruding boxes are leaf nodes.

Each node in a level is represented as a bar (or box). The width encodes the number of immediate children, and the hue encodes the total children. This encoding scale is only relative to each respective level. That is, the widest box in level only has the most children at that level. And similarly, the reddest box only has the most children for that specific level. From this, we can see

whether the paths to a node passes through a landmark in that level, or whether the parents along the path have many other children.

4.6 Search, Pin, and Details

Outside the exploration loop (i.e. V1 – V5), users can search and save (or pin) individual terms in the ontology. Each keystroke entered by the user automatically updates the results in V7 (see Figure 19 and Figure 31). Consequently, this result set is automatically brushed onto V1. In this scenario, V1 acts as a map to show the distribution of the search results. In Figure 31, the search results for 'heart' are represented as white circles on top of V1. Each result's location is mapped by their major subtree (slice) and their level. From this representation, we can see that most of the 'heart' results reside under the slice 'Cardiovascular System', as expected. Clicking on a search result node will further update the remaining visualizations (i.e. V2 – V5).

Figure 31 Searching and Pinning items. Pinned items are brushed throughout all visual representations.

Search results can also be pinned or unpinned by clicking the 'pin', or 'unpin' button beside it, respectively (see Figure 31). The purpose of pinning is to show user specified items in the ontology alongside other landmarks presented by the other visualizations. Currently, we allow a user to pin up to 5 items. These 5 items are encoded with the 5 most distinct colours (we show 4 in Figure

31). These distinct colours act as visual cues to discern pinned items from other items. In other words, pinned items are brushed onto the other visualizations (see Figure 31). That is, in all visualizations, if a node is pinned, its colour is changed to the pinned colour.

Lastly, linking together with V5 is V6 details panel. Here, all original properties of an ontology entity are displayed (e.g. ID, definition, synonym, etc.). This panel is shown in Figure 32 (left). In addition, this panel acts as a linking point to other resources that maybe use this ontology. In our example, we can link an explored or searched item back to HPO's own browser (Figure 32).

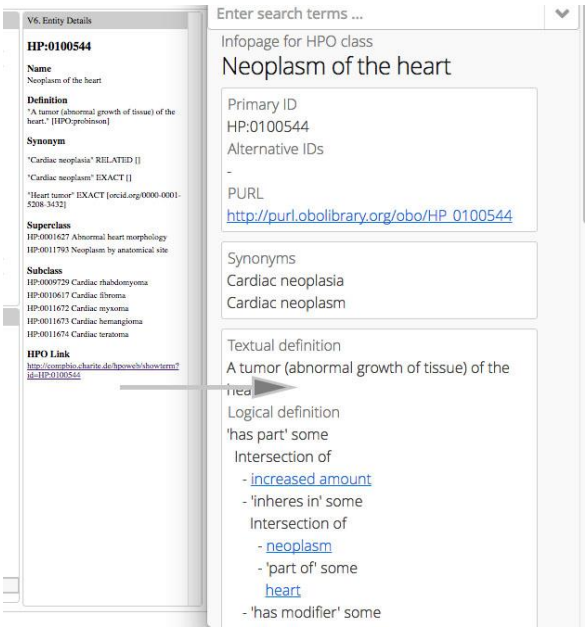


Figure 32 V6) Details panel linking to HPO-Browser

4.7 Implementation

The latest version of MOE resides at http://linjoey.github.io/MOE/ as a web application. Chrome is recommended for viewing.

This prototype is implemented in 2 stages: the preprocessing stage, and the visualization stage. The preprocessing stage is concerned with processing an ontology file to be used in the visualizations. The preprocessing involves computing the transitive closure (i.e. the total children) of every node. This value is used as an indicator for landmarks. To achieve this, we use python

and the library NetworkX[23]. First, we parse the ontology file into a NetworkX graph and apply the algorithm *descendants* [24] (transitive closure) on each node. Then, we output this graph as a JSON format to be used with JavaScript.

In the visualization stage, we use the JavaScript visualization library D3.js[25]. D3 provides facilities to effectively map the processed data into visual representations. In addition, D3 provides convenience methods to implement complex interactions such as zooming and dragging.

Each visualization, i.e., V1 to V5, is implemented following the MVC (model-view controller) design pattern. We separate the data model (ontology) from the view (visual representations, and the controller (interaction) code. This separation is encapsulated in an object-oriented class in JavaScript, i.e., each class implements a visualization. Each visualization class requires a data source as input, a method to draw the data into visual representations, and a method to update the visual representations upon user input (e.g. drag or click). In addition, each visualization can emit events (such as on user click) in which other visualization can listen to—this allows synchronous update and dynamic brushing of multiple visualizations.

[23] https://networkx.github.io/
[24] http://networkx.readthedocs.io/en/stable/reference/generated/networkx.algorithms.dag.descendants.html?highlight=descendants
[25] https://d3js.org/

Chapter 5 Conclusions

5.1 Summary

An ontology is an explicit specification for a domain of discourse. This specification includes a set of standardized vocabulary (or terms, classes, entities) and a set of semantic relationships among these terms. The relationships include *is-a* inheritance or other domain specific relationships, e.g. part-of, or connected-to. In its most basic form, ontologies have a hierarchical structure through the is-a relations; however, due to the possibility multiple inheritance, this structure is not strictly a tree, but a directed graph. Ontologies are represented as a set of triples in the form of *object-property-value*. This triple can describe different aspect of the ontology, e.g. the inheritance tree. This list of triple is usually encoded as an OWL (Web Ontology Language) or OBO (Open Biomedical Ontologies), among others, file specification for machine reading.

Ontologies have applications in various fields and industries. Some examples include the following: using an ontology in embryology to keep the standardization terms consistent with visualization software; and connecting various databases in a marine biology context, e.g. fish database connected with geography database.

In the field of ontology engineering, authoring and maintaining large ontologies can be a difficult task—particularly when the size of ontologies can be in the tens of thousands of terms. Many ontology authoring tools such as Protégé and NeOn include visualization methods to alleviate this complexity. In past works, tree and graph based visualization and interaction techniques have been applied to ontologies, some of these styles include: node-link graphs, space-filling techniques, zoomable interfaces, 3D interfaces, and focus+context distortions. However, each of these visualization techniques have their own individual shortcomings. For instance, the treemap (a space-filling technique) is effective at showing different leaf node attributes, but it is confusing for showing hierarchical relationships; yet a node-link graph is good at showing relationships but it is also very space inefficient in terms of screen real-estate. However, regardless of the visualization technique, the huge size of ontologies make it futile for any single visualization style to visualize the entire ontology. Any attempt to visualize the entire ontology will result in screen clutter and occlusion.

In this thesis, we take a multi-view approach to visualizing ontologies. In each view, we break up the ontology into 5 different levels of abstraction (collectively named V1 to V5). These different levels of abstractions start from a high-level overview down to a low-level entity. In each view, we design novel visual representations and use interaction techniques to support the exploration and navigation of ontologies. In addition, we support the formation of mental models by encoding landmarks, routes, and survey knowledge. In this prototype, we designate the total connected children (or the transitive closure of a node in a graph), as a landmark value. That is, a node with more connected nodes is considered more important at that level of abstraction. Lastly, our design incorporates search and pin features to better support the sense-making process (V6 and V7).

The 5 visual representations (V1 – V5) visually encode the different levels of abstractions as follows.

V1: Global Overview. V1 provides an overview of the entire ontology from the root. This overview provides a sense of landmarks (or subtrees, subsections) at a very high level. In addition, this overview provides a sense of the overall structure of the ontology, i.e., the depth and the breadth. Finally, this overview acts as a global map where search results are brushed onto. V1 attempts to answers the following questions when first exploring the ontology:

1. Around how many major landmarks (subtrees) does the ontology have at the root?
2. Among the top landmarks, which one is the deepest (i.e. level), and which one is the shallowest?
3. Among the top landmarks, how many nodes are located at each level? Alternatively, which levels have the most nodes? And which have the least?
4. How can a user quickly see the location of an entity in the entire ontology?

V2: Subsection Levels. V2 provides more details for each major subsection in V1. In V2, an overview of individual levels is represented. V2 attempts to answer the following questions regarding a selected subsection from V1:

1. What are the major landmarks at each level?
2. How does the landmark at each level relate to other nodes? That is, are there many other landmarks around it or just one?

3. What is the general distribution of nodes at each level in terms of the landmark value?
4. Which parents contribute to the nodes at each level?
5. Are the contributing parents close or far away? That is, are they mostly from the level above, or higher?
6. Are specific nodes at a level from the same or different parents?

V3: Landmark Distances. V3's scope is within a single level of a major subsection. This level is directly connected with a level in V2. V3 is concerned with representing the exact distances among top landmarks in level. V3 attempts to answer the following questions when a level is selected in V2:

1. What are the names of the top 13 or so landmarks within this level?
2. How does the top 2-3 landmarks compare to the top 10-13 landmarks? That is, are they similar, or differ largely in the landmark value?
3. What are the distances among these top landmarks?
4. Is there a cluster of landmarks that are close together?
5. Are the top landmarks far away?

V4. Entity Network and **V5. Path Explorer.** V4 and V5 is scoped to two entities and a single entity, respectively. V4 and V5 provides the fine details that was abstracted away in V1 to V3. V4 shows the immediate parents and children of a single node, along with the common parents if two nodes are selected. And V5 shows all the available paths to reach a node. Collectively, V4 and V5 attempts to answer the following questions when one or two entities is selected:

1. Which entities are the parents and children of these selected node(s) and around what levels do they reside in?
2. What is the common parent among the two selected nodes?
3. What are the paths to reach the common parent from the two selected nodes?
4. What are all the paths from the root that reaches the selected node?
5. Are the elements along the path a landmark at its own level?

V6 Details and **V7 Search and Pin.** V6 is a simple list of all the details originally provided in the ontology. This list of details includes extra metadata such as definition, synonyms, and the ID,

among others. Finally, V7 provides a simple interface to support search and saving of entities. V6 and V7 attempts to support the following tasks:

1. Show the full details of a term in the ontology.
2. Show external links that an entity may be connected to.
3. Search for terms containing the entered keywords.
4. Save terms for later use
5. Brush search and saved results onto other visual representations.

5.2 Comparison of MOE with Other Ontology Visualization Tools

In this section, we outline the key differences and contributions of MOE compared to the previously discussed ontology visualization methods and tools. Here, we examine the differences of VR design, presentation design, interaction design, and the high-level approach to visualizing large ontologies.

MOE incorporates various new and novel VR techniques. These techniques were iteratively designed with the VR pattern framework by (Sedig & Parsons, 2015) outlined in section 3.2. V1 and V2 contain novel VR techniques used to address the abstraction constraints of ontologies at a high level. V1 uses an aggregate approach to summarize the entire ontology into a single cell-based table representation. Here, V1 is not concerned with individual entities, but rather higher level concepts such as major subtrees and levels. In contrast, traditional ontology visualization methods (e.g. treemaps, node-link graphs, indented-list) encode for individual entities and links.

Similarly, V2's VR approach is to avoid the representation of individual entities at a level, but rather to show a higher-level concept, that is, the key landmarks at each level relative to other nodes in the same level. A novel aspect of V2 is its ability to show each level's contributing parents. This abstraction idea is not seen in previous tools: e.g. tools that implement a node-link graph technique show the immediate parents of current focused nodes via a link, but, how those parents relate to one another, what their landmark values are, or their depth, is not encoded. V1 and V2 both focus on a high-level idea of an ontology, and we design novel solutions to support these ideas.

The related work KC-Viz (Motta, 2012) contains similar goals to this thesis, i.e., visualizing large ontologies while supporting the task of sense-making. Like KC-Viz, MOE borrows the idea of encoding 'key-concepts', or landmarks, of the ontology; support basic exploration features such as search and save; and interaction techniques such as collapsing or expanding nodes. However, a key difference between MOE and KC-Viz is the general approach to the exploration process. KC-Viz incorporates a 'middle-out' approach, i.e., by first providing key-concepts of the entire ontology first, and allow the users to actively expand or collapse paths or nodes. In comparison, MOE separates levels of conceptual abstraction of the ontology, and allow the user to start from a high-level idea down to a low-level entity. An advantage of this high-level to low-level approach is its ability to provide contextual information such as 'how did we get here?', or 'where can we go from here?'. In the context of ontologies, it may be disorienting if a user is presented with a deeply nested item immediate without first providing the higher-level paths, e.g. the major subtree it's from, the depth, the key landmarks it passes, and so on.

The idea of using multi-views for ontology visualization is comparable to the works of (Dmitrieva & Verbeek, 2009). In their approach, a specific area of an ontology can be visualized in different geometric views by different filters. However, a difference in MOE is its ability to connect multiple views in one place, and connect them with dynamic linking. An advantage of this approach is to see all different levels of abstraction at the same time, and see how changes in one view apply to the others.

A limitation of MOE is its lack of encoding for ontology instances; and a focus of instance visualization is seen in OntoTrix (Bach et al., 2011). Although instances are part of the ontology specification, many ontology visualization tools (e.g. Protégé plugin IsaViz) omit this feature. The authors of OntoTrix stress the potential importance of ontology instances, as in some cases, instances can make up the bulk of the ontology. A potential next step with MOE is to integrate more levels of abstractions, i.e. more connected VRs that encode for ontology instances.

In summary, MOE draws together many favorable ideas of various ontology visualization tools. Some of these ideas include: using multiple views to show different aspects of the ontology; support the formation of mental models; use classic representations (e.g. node-link graphs); and implement traditional visualization techniques (e.g. showing overview and details, and

focus+context). In addition, MOE takes a novel approach of separating the ontology exploration process into five levels of abstraction; use new and novel VRs to visualize high-level ideas; and connect the idea of encoding for landmarks, route, and survey knowledge to support the formation of mental models throughout all levels of abstraction.

5.3 Future Work

The prototype in its current form still lacks some advanced features. For instance, the search panel can be incorporated with dynamic query to better support the filtering of results. These filters can filter properties such as children count, parent count, or the landmark value, among others. Another feature lacking is the ability to load ontology files on-the-fly and visualize them instantly. Currently, a prepressing stage is required on a local machine before visualizing new ontologies. This may be an annoyance for expert users that need to visualize many ontologies. Outside the prototype, some areas to explore include the visualization of multiple ontologies, and the visualization of instances.

References:

Amann, B., & Fundulaki, I. (1999). Integrating ontologies and thesauri to build rdf schemas. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *1696*, 234–253. https://doi.org/10.1007/3-540-48155-9_16

Antoniou, G., Groth, P., Van Harmelen, F., & Hoekstra, R. (2012). *A Semantic Web Primer*. *The effects of brief mindfulness intervention on acute pain experience: An examination of individual difference* (Vol. 1). https://doi.org/10.1017/S0269888909990117

Bach, B., Pietriga, E., Liccardi, I., Legostaev, G., Bach, B., Pietriga, E., … Liccardi, I. (2011). OntoTrix : A Hybrid Visualization for Populated Ontologies To cite this version : OntoTrix : A Hybrid Visualization for Populated Ontologies.

Bertin, J. (1981). Graphics and graphic information-processing. In *Graphics and graphic information-processing* (pp. 24–31). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0019654060&partnerID=40

Bikakis, N., & Sellis, T. (2016). Exploration and visualization in the web of big linked data: A survey of the state of the art. *CEUR Workshop Proceedings*, *1558*.

Bizer, C., Heath, T., Idehen, K., Berners-lee, T., Mit, W. C., Berlin, F. U., & Software, O. (2008). Linked data on the web. *WWW2008 Workshop on Linked Data on the Web*, 1265–1266. https://doi.org/10.1145/1367497.1367760

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). Readings in Information Visualization: Using Vision to Think. In *Information Display* (Vol. 1st, p. 686). Retrieved from http://portal.acm.org/citation.cfm?id=300679

Chernoff, H. (1973). The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association*, *68*(342), 361–368. https://doi.org/10.1080/01621459.1973.10482434

Dhillon, S. K., Chiew, S. L., Leow, L. K., Sidhu, A. S., Shuhaimi, N. I., Leong, Y. M., & Chong, V. C. (2013). A model of a digital biological ecosystem. *Systematics and Biodiversity*, *11*(4), 425–435. https://doi.org/10.1080/14772000.2013.856962

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). Human-Computer Interaction. *Human-Computer Interaction*, *Third*(January), 834. https://doi.org/10.1207/S15327051HCI16234

Dmitrieva, J., & Verbeek, F. J. (2009). Multi-view ontology visualization. *The 11th International Protégé Conference*, 1–4. Retrieved from http://protege.stanford.edu/conference/2009/abstracts/S9P3Dmitrieva.pdf

Eco, U. (1977). A Theory of Semiotics (review). *The Journal of Aesthetics and Art Criticism*, *35*(44), 476–478. https://doi.org/10.1016/0024-3841(77)90016-X

Fekete, J., Wijk, J. Van, Stasko, J., North, C., Fekete, J., Wijk, J. Van, … Value, T. (2012). The Value of Information Visualization To cite this version : The Value of Information Visualization.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, *5*(2), 199–220. https://doi.org/http://dx.doi.org/10.1006/knac.1993.1008

Gutwin, C., & Fedak, C. (2004). A Comparison of Fisheye Lenses for Interactive Layout Tasks. *Proceedings of Graphics Interface*, 53–60.

Hansen, C., & Johnson, C. (2004). *The Visualization Handbook*. *The Visualization Handbook*. https://doi.org/TK7882.I6V59

Henry, N., Fekete, J., & Mcguffin, M. J. (2007). NodeTrix : A Hybrid Visualization of Social Networks, (March).

Johnson, B. (1992). TreeViz : Treemap Visualization of Hierarchically Structured Information. In *Proceedings of theSIGCHI conference on Human Factors in Computing Systems* (pp. 369–370). https://doi.org/10.1145/142750.142833

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods - A survey. *Acm Computing Surveys*, *39*(4), 10. https://doi.org/http://doi.acm.org/10.1145/1287620.1287621

Katifori, A., Torou, E., Vassilakis, C., Lepouras, G., & Halatsis, C. (2008). Selected results of a comparative study of four ontology visualization methods for information retrieval tasks. *2008 Second International Conference on Research Challenges in Information Science*, (September 2016), 133–140. https://doi.org/10.1109/RCIS.2008.4632101

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, *8*(1), 1–8. https://doi.org/10.1109/2945.981847

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science*, *18*(4), 513–549. https://doi.org/10.1016/0364-0213(94)90007-8

Lamping, J., Rao, R., & Pirolli, P. (1995). A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95* (pp. 401–408). New York, New York, USA: ACM Press. https://doi.org/10.1145/223904.223956

Lanzenberger, M., Sampson, J., & Rester, M. (2010). Ontology Visualization: Tools and Techniques for Visual Representation of Semi-Structured Meta-Data. *Journal Of Universal Computer Science*, *16*(7), 1036–1054. Retrieved from http://www.jucs.org/jucs_16_7/ontology_visualization_tools_and/jucs_16_07_1036_1054_lanzenberger.pdf

Mackinlay, J. D., Robertson, G. G., & Card, S. K. (1991). The perspective wall: detail and context smoothly integrated. *CHI '91: Proceedings of the SIGCHI Conference on Human Factors in*

Computing Systems, (December), 173–176. https://doi.org/10.1145/108844.108870

Meyer, R. (2013). Knowledge Visualization Currents. *Trends in Information Visualization*, *23*, 63–84. https://doi.org/10.1007/978-1-4471-4303-1

Motta, E. (2012). Visualizing and Navigating Ontologies with KC- Viz, (January). https://doi.org/10.1007/978-3-642-24794-1

Norman, D. A. (1988). The Psychology of Everyday Things. *The Psychology of Everyday Things*, 1–104. https://doi.org/10.2307/1423268

Noy, N. F., & McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*, 25. https://doi.org/10.1016/j.artmed.2004.01.014

Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R. W., & Musen, M. A. (2001). Creating semantic web contents with protégé-2000. *IEEE Intelligent Systems and Their Applications*, *16*(2), 60–71. https://doi.org/10.1109/5254.920601

Parsons, P., & Sedig, K. (2014). Common Visualizations: Their Cognitive Utility. *Handbook of Human Centric Visualization*, 671–691. https://doi.org/10.1007/978-1-4614-7485-2

Parsons, P., Sedig, K., Didandeh, A., & Khosravi, A. (2015). Interactivity in visual analytics: Use of conceptual frameworks to support human-centered design of a decision-support tool. In *Proceedings of the Annual Hawaii International Conference on System Sciences* (Vol. 2015–March, pp. 1138–1147). https://doi.org/10.1109/HICSS.2015.138

Plaisant, C., Grosjean, J., & Bederson, B. B. (2002). SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings - IEEE Symposium on Information Visualization, INFO VIS* (Vol. 2002–Janua, pp. 57–64). https://doi.org/10.1109/INFVIS.2002.1173148

Rabattu, P.-Y., Massé, B., Ulliana, F., Rousset, M.-C., Rohmer, D., Léon, J.-C., & Palombi, O. (2015). My Corporis Fabrica Embryo: An ontology-based 3D spatio-temporal modeling of human embryo development. *Journal of Biomedical Semantics*, *6*(1), 36. https://doi.org/10.1186/s13326-015-0034-0

Robertson, Goerge G.; Mackinlay, Jock D.; Card, S. K. (1991). Cone-Trees: Animated 3D Visualizations of Hierarchical Information. *Proceeding CHI '91 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/108844.108883

Sagha, A. (2014). Graph-Based Representation and Reasoning, *8577*, 204–221. https://doi.org/10.1007/978-3-319-08389-6

Scaife, M., Rogers, Y., & SC, M. (1996). External cognition: how do graphical representations work? *International Journal*, *45*(2), 185–213. https://doi.org/10.1006/ijhc.1996.0048

Sedig, K., & Parsons, P. (2013). Interaction Design for Complex Cognitive Activities with Visual Representations: A Pattern-Based Approach. *AIS Transactions on Human-Computer Interaction*, *5*(2), 84–133. https://doi.org/10.5121/ijfcst.2014.4403

Sedig, K., & Parsons, P. (2015). Design of Visual Representations for Human-Information Interaction : A Pattern-Based Framework, 1–117.

Sedig, K., Parsons, P., & Babanski, A. (2012). Towards a Characterization of Interactivity in Visual Analytics. *Journal of Multimedia Processing and Technologies, Special Issue on Theory and Application of Visual Analytics*, *3*(1), 12–28. https://doi.org/10.1145/0000000.0000000

Sedig, K., Parsons, P., Dittmer, M., & Haworth, R. (2013). Human-Centered Interactivity of Visualization Tools: Micro- and Macro-level Considerations. In *Handbook of Human Centric Visualization* (pp. 717–743). https://doi.org/10.1007/978-1-4614-7485-2_29

Shneiderman, B. (1996). The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations, 336–343.

Spence, R. (2002). *Information Visualization*. *Information Visualization* (Vol. 1). https://doi.org/10.1057/palgrave/ivs/9500009

Spence, R. (2007). *Information Visualization: Design for Interaction*. *Proceedings of CHI 2005 Conference on Human Factors in Computing Systems* (Vol. 7). https://doi.org/10.1075/idj.17.3.15ehr

Spence, R. (2014). *Information Visualization*. *Wiley Interdisciplinary Reviews: Computational Statistics* (Vol. 2). https://doi.org/10.1007/978-3-319-07341-5

Suárez-Figueroa, M. C., García-Castro, R., Villazón Terrazas, B., & Gómez-Pérez, A. (2011). Essentials in Ontology Engineering: Methodologies, languages and tools. *Proceedings of the 2nd Workshop Organized by the Eeb Data Models Community- CIB Conference W078-W012*, 9–21.

Suárez-Figueroa, M. C., Gómez-Pérez, A., & Fernández-López, M. (2012). The neon methodology for ontology engineering. In *Ontology Engineering in a Networked World* (pp. 9–34). https://doi.org/10.1007/978-3-642-24794-1_2

Tominski, C. (2015). *Interaction for Visualization*. *Synthesis Lectures on Visualization* (Vol. 3). https://doi.org/10.2200/S00651ED1V01Y201506VIS003

Tominski, C., Gladisch, S., Kister, U., Dachselt, R., & Schumann, H. (2016). Interactive Lenses for Visualization : An Extended Survey, *0*(0), 1–28. https://doi.org/10.1111/cgf.12871

Tversky, B. (1991). Spatial Mental Models. *Psychology of Learning and Motivation - Advances in Research and Theory*, *27*(C), 109–145. https://doi.org/10.1016/S0079-7421(08)60122-X

Tversky, B. (1993). Cognitive Maps, Cognitive Collages, and Spatial Mental Models. *Spatial*

Information Theory: A Theoretical Basis for GIS, Proceedings COSIT'93, 14–24. https://doi.org/10.1007/3-540-57207-4_2

Van Wijk, J. J., & Van de Wetering, H. (1999). Cushion treemaps: visualization of hierarchical information. In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)* (p. 73–78,). https://doi.org/10.1109/INFVIS.1999.801860

Wang, X., & Almeida, J. S. (2007). Techniques for ontology visualization. In *Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences* (pp. 185–203). https://doi.org/10.1007/978-0-387-48438-9_10

Ware, C. (2004). *Information visualization: perception for design*. *Information Visualization*. https://doi.org/10.1016/B978-0-12-381464-7.00018-1

Ware, C. (2012). *Foundation for a Science of Data Visualization*. *Information Visualization: Perception for Design*. https://doi.org/10.1016/B978-0-12-381464-7.00001-6

White, D. R., & Borgatti, S. P. (1994). Betweenness centrality measures for directed graphs. *Social Networks*, *16*(4), 335–346. https://doi.org/10.1016/0378-8733(94)90015-9

Appendices

Appendix A: Visual Representation Pattern Framework

Table 1 Visual representation pattern framework, adapted from (Sedig & Parsons, 2015)

Pattern	Characterization	Code
Token	Map information items onto a single unitized VR. Most basic pattern for representing information items. Most VRs are based on a blending of token with other patterns. Can be useful when wanting to represent an item, such as an object, property, value, system, operation, location, direction, feature, function, or process, in an individual, unitized fashion in the representation space. Instantiations are often small.	TK
Area	Map information items onto VRs in such a way that their boundary, shape, region, and/or area are encoded. Can be useful when wanting to represent information items that have two or three spatial dimensions with a definite shape, boundary, or region.	AR
Cell	Map information items onto VRs and organize them by segmenting, compartmentalizing, or containing them within cell-like structures. Can be useful when wanting to create partitions or compartments for a set of VRs according to their temporal, ordinal, nominal, categorical, conceptual, or spatial properties, and want to represent distributed membership, containment, division, or separation of information.	CL
Coordinate	Map information items onto VRs and organize them with respect to a frame of reference. Can be useful when wanting to represent information items that can be placed and located with respect to external structures. Typically instantiated with axes or scales, which allow for describing the location or position of VRs with respect to such structures.	CR
Branch	Map information items onto VRs and organize them in the representation space in a branched and/or subdivided fashion. Can be useful when wanting to represent converging and/or diverging (e.g., temporal, categorical, conceptual, or geographic) features of information items in addition to whether they originate from or terminate in other sources.	BR
Cycle	Map information items onto VRs and organize them in a circular, wheel-like, rotational, spiral, and/or cyclical fashion. Can be useful when wanting to represent information items that are periodic and have recurrence. Using a circular structure in a VR does not necessarily mean that it is based on the cycle pattern.	CC
Fusion	Map a number of information items onto a single VR in a continuous fashion, such that they are integrated and fused together. Can be useful when wanting information items to be represented in such a way that their conjoinment makes them indistinguishable from one another.	FS
Group	Map information items onto VRs and organize them by congregating them close to each other. Can be useful when wanting to represent proximate features of information items—whether be semantic, syntactic, spatial, temporal, functional, or otherwise.	GR

Hierarchy	Map information items onto VRs and organize them in a hierarchical, multi-level, pyramid-like fashion, where higher levels are superior to or contain and encompass lower level VRs. Can be useful when wanting to represent parent-child, higher-lower, superior- subordinate, whole-part, container-contained, or system-part relationships.	HR
Link	Map information items onto VRs and organize them by connecting using paths, routes, lines, or other similar structures. Can be useful when wanting to represent explicit connections, relationships, and/or associations between and among VRs, whether temporal, causal, functional, conceptual, epistemological, or otherwise.	LK
List	Map information items onto VRs and organize them by placing in a sequential, successive fashion. Can be useful when wanting to represent information items that have a definite order—whether chronological, alphabetical, or otherwise—as well as a clear start and end point.	LS
Spectrum	Map information items onto VRs and organize them in a spectral fashion. Can be useful when wanting to represent variability within a VR or across a number of VRs, such as when representing different densities, temperatures, incomes, and ratios. Often instantiated using multiple saturation or luminance values of a particular hue, or using multiple hues or textures.	SP
Stack	Map information items onto VRs and organize them by placing on-top of one another in a piled or stacked fashion. Can be useful when wanting to represent information items that have similar, co-occurrent, or co-existent features. VRs are often placed on-top of one another such that they are touching or are very close together.	ST
Track	Map information items onto VRs and organize them in a lane-, stripe-, and/or track-like fashion. Can be useful when wanting to represent information items that exist or occur within routes, paths, or channels, whether conceptually, spatially, temporally, or otherwise. Also sometimes useful when items have co-occurrence, co-existence, simultaneity, and/or synchrony.	TR

Appendix B: EDIFICE-AP Catalogue of epistemic action patterns

Table 2 EDIFICE-AP Epistemic Action Patterns, adapted from (Sedig & Parsons, 2013)

	Action	**Characterization**
Unipolar	Annotating	augment them with additional visual marks and coding schemes, as personal meta-information
	Arranging	change their ordering, either spatially or temporally
	Assigning	bind a feature or value to them (e.g., meaning, function, or behavior)
	Blending	fuse them together such that they become one indivisible, single, new VR
	Cloning	create multiple identical copies
	Comparing	determine degree of similarity or difference between them
	Drilling	bring out, make available, and display interior, deep information
	Filtering	display a subset of their elements according to certain criteria
	Measuring	quantify some items (e.g., area, length, mass, temperature, and speed)
	Navigating	move on, through, and/or around them
	Scoping	dynamically work forwards and backwards to view compositional development and growth
	Searching	seek out the existence of or locate position of specific items, relationships, or structures
	Selecting	focus on or choose them, either as an individual or as a group
	Sharing	make them accessible to other people
	Transforming	change their geometric form
	Translating	convert them into alternative informationally or conceptually-equivalent forms
Bipolar	Accelerating/ Decelerating	increase or decrease speed of movement of their constituent components
	Animating/ Freezing	generate or stop motion in their constituent components
	Collapsing/ Expanding	fold in or compact them, or oppositely, fold them out or make them diffuse
	Composing/ Decomposing	assemble them and join them together to create a new, whole VR, or oppositely, break whole entities up into separate, constituent components
	Gathering/ Discarding	gather them into a collection, or oppositely, throw them away completely
	Inserting/ Removing	interject new VRs into them, or oppositely, get rid of their unwanted or unnecessary portions
	Linking/ Unlinking	establish a relationship or association between them, or oppositely, dissociate them and disconnect their relationships
	Storing/ Retrieving	put them aside for later use, or oppositely, bring stored VRs back into usage

Appendix C: EDFICE-IVT micro level considerations

Table 3 EDIFICE-IVT micro level considerations, adapted from (Sedig et al., 2013)

Component	Element	Concern	Forms
Action	Presence	Existence and advertisement of action	Explicit, implicit
	Agency	Metaphoric agency through which action is expressed	Verbal, manual, pedal, aerial
	Granularity	Constituent steps of action	Atomic, composite
	Focus	Focal point of action	Direct, indirect
	Flow	Parsing of action in time	Discrete, continuous
	Timing	Time available for user to compose and/or commit action	User-paced, system-paced
Reaction	Activation	Point at which reaction begins	Immediate, delayed, on-demand
	Flow	Parsing of reaction in time	Discrete, continuous
	Transition	Presentation of change	Stacked, distributed
	Spread	Spread of effect that action causes	Self-contained, propagated
	State	Condition of VRs as interface reaches equilibrium	Created, altered, deleted
	Context	Context in which VRs exist as interface reaches equilibrium	Changed, unchanged

Curriculum Vitae

Name:	Zhao Lin
Post-Secondary Education and Degrees	The University of Western Ontario London, ON, Canada 2015 – 2017 M.Sc. Computer Science The University of Western Ontario London, ON, Canada 2010 – 2015 B.Sc. Computer Science
Related Work Experience	Teaching Assistant The University of Western Ontario 2015 – 2017 Software Engineer Hewlett-Packard Company 2013 – 2015
Honours and Awards	Western Graduate Research Scholarship 2015 – 2017