Encounters in Theory and History of Education
Vol. 19, 2018, 163-185

Compara

Marcus A. Gordon, Michael Carnevale, Minsheng Zheng, Dr. Sara Diamond
Visual Analytics Lab, OCAD University, Toronto, Canada

Abstract: The OCAD Visual Analytics Laboratory (VAL) has developed a taxonomy of end users, software systems, data types, tasks and interactivity within the domain of smart city transportation planning. This paper contributes to the taxonomy by creating Compara, an intuitive, interactive and searchable index that visualizes the attributes of software from a wide-range of applications and technologies. The taxonomy began as a spreadsheet that we transformed into a custom interactive data visualization that could help users find and understand existing tools and their attributes

The taxonomy and interactive index are a component of the iCity project which brings together academic, government, and industrial partners in order to improve the quality of life for urban residents and visitors through the development and integration of advanced IT infrastructure for the purpose of managing transit and transportation. The taxonomy and resulting tools discussed in this paper have expanded to include resources for urban planning which also impact transit and transportation. One of the primary uses of the taxonomy is to help various iCity project teams and stakeholders locate software that may be useful to their design and development process, as well as to understand the end-users of this technology. The taxonomy may be used to develop a city management dashboard for city planners and analysts, or equally, to design a city-services facing mobile service application.

Keywords: Smart Cities, Interface design, urban planning, Data visualization, taxonomy

Résumé: Le Laboratoire d'Analyse Visuelle de OCAD (VAL) a développé une taxinomie d'utilisateurs finals, de systèmes de software, types de données, tâches et activités dans le domaine de la planification du transport des villes intelligentes. Ce papier contribue à une taxinomie en créant Compara, un index intuitif, interactif et de recherche qui visualise les caractéristiques de software à partir d'une large gamme d'applications et de technologies. La taxinomie débuta par une feuille de calcul que nous avons transformée en visualisation interactive de données personnalisées pour aider les utilisateurs à trouver et comprendre les outils disponibles et leur fonction. La taxinomie et son index interactif sont un élément du projet iCity qui relie les partenaires académiques, gouvernementaux et industriels pour améliorer la qualité de vie des résidents urbains et des visiteurs en développant et intégrant les infrastructures de TI pour mieux gérer le transit et le transport. La taxinomie et les outils qui en résultent et sont analysés dans ce document se sont amplifiés pour inclure des ressources pour la planification urbaine qui ont également un impact sur le transit et le transport. La taxinomie est utilisée principalement pour aider les différentes équipes de projets iCity, et les partis

ISSN 2560-8371
Date of submission: 2018-06-19 http://doi.dx.org/ 10.24908/eoe-ese-rse.v19i0.11867
Date of acceptance: 2018-11-14

intéressés, à trouver le software pouvant être utile à leur processus de design et de développement, ainsi qu'à comprendre les destinataires de cette technologie.

La taxinomie peut être utile pour aider à développer un panel d'administration pour les planificateurs de villes et les analystes, ainsi que pour aider à dresser un plan de services urbains mobiles pour la ville en quête d'une application de tels services.

Mots-clés: Les villes intelligentes, Les indices interactifs, La planification urbaine, Visualisation de données, Taxonomie

Resumen: El Laboratorio de Analítica Visual (VAL) del Ontario College of Arts and Design (OCAD) ha desarrollado una taxonomia de usuarios finales, sistemas de software, tipos de datos, tareas e interactividad dentro del dominio del plan de transportation de la ciudad inteligente. Este artículo contribuye a una taxonomía con la creación de Compara, un index intuitivo, interactivo e investigable que visualiza los atributos del software desde un amplio espectro de aplicaciones y tecnologías. La taxonomía comenzó como una hoja de cálculo (spreadsheet) que transformamos en visualización interactiva personalizada de datos que puede ayudar a los usuarios a encontrar y entender herramientas (tools) y atributos existentes.

La taxonomía y el índice interactivos son un componente del proyecto iCity que une a miembros de la academia, el gobierno, y la industria con el objeto de mejorar la calidad de vida de los residentes urbanos así como de los visitantes a través del desarrollo e integración de una infrastructura IT avanzada con el propósito de gestionar (Managing) tránsito y transportación. Uno de los usos primarios de la taxonomía es ayudar a various equipos y a las partes interesadas del proyecto iCity a localizar software que pueda ser útil para sus procesos de diseño y desarrollo, asi como entender a los usuarios de esta tecnología. La taxonomía puede ser usada para desarollar un panel de gestión para los planificadoresy analistas de la ciudad o igualmente para diseñar servicios móviles de la ciudad.

Palabras claves: ciudades inteligentes, diseño interfaz (interface), planificación urbana, visualización de información, taxonomía

Editor's note:

This is a pdf version of a digital native work. For a fuller appreciation of the work, we encourage you to view the html version, which is currently housed at: https://ojs.library.queensu.ca/public/journals/6/content/gordon/index.html.

This work references an interactive web-based visualization that the reader is encouraged to explore. The interactive web-application is currently housed at: https://ojs.library.queensu.ca/public/journals/6/content/gordon/app.html.

Description of Visualization Goals

The data visualization problem we address is how to convert a large and growing spreadsheet of categorized, hierarchical, and relational qualitative data into an interactive visual diagram that can be used to quickly understand the broad-scale application landscape of transit and transportation digital tools. We look for hierarchical patterns and represent the associations that the taxonomy indicates of sub-attributes. Put another way, how can we visualize a software taxonomy and the relationships that it captures? It includes sub-attributes such as user-types, technologies, data types and format, tasks, visualization types, and levels of interactivity. The aim of the taxonomy is to help end-users sort through the software landscape to understanding the qualities of domain tools and then choose products or design software to meet the needs of the domain. To be successful, Compara must deploy a consistent glossary of terms, so that attribute terms appear for multiple software products. Compara must also allow users to identify software associated with specific user-types. Since user-type is one attribute amongst many, Compara must generalize the identification algorithm for other attributes. The scope and limitations of this data visualization problem is impacted by the reality that the software taxonomy can never said to be fully complete as new software are always being released, and identifying every last one in the market is difficult. This is a data collection problem not a visualization problem. Ideally, the visualization that we develop should be able to generalize to other instantiations of this kind of problem, namely visualizing an entire hierarchical index and its associated attributes. New data can constantly be added.

Figure 1 illustrates the overall structure of the taxonomy spreadsheet in its original instantiation.

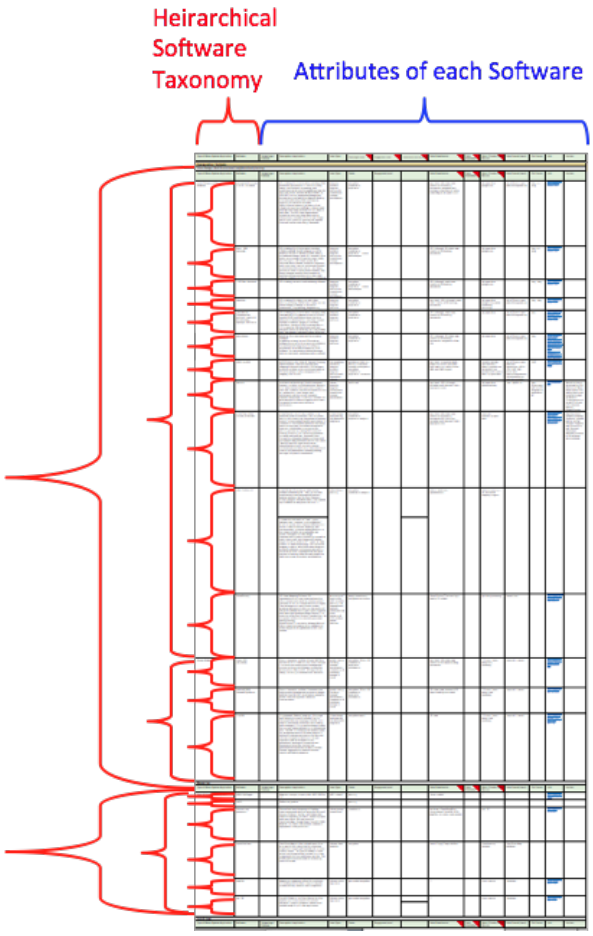

Figure 1: Cropped subsection of spreadsheet showing how the software hierarchy is organized, as well as the associated attributes for each software. Software are categorized based on their use and type. Smallest brackets represent individual software.

Objectives and Questions

The immediate objectives and research questions are:

- To implement a visualization that will simultaneously show an hierarchical qualitative structure, as well as its associations to external attributes. Does such a visualization technique exist?
- To determine if the spreadsheet format can be simplified to show the overall structure of its relational contents more clearly.
- To investigate how an indexical structure can be visualized in an aesthetic and engaging way.
- To investigate whether this visualization can be implemented most effectively as a static visual artifact, or if its functional purpose can be improved through interactive elements?
- To determine how a visualization technique for Compara can be generalized and scaled as the software taxonomy continues to grow.

Literature Review

We focused on the general themes of hierarchical structures, node-link representation, and examples of visual taxonomy from software research. A taxonomy is essentially a system of classification that typically organizes categories by their sub-ordinate or super-ordinate relationships within a hierarchy (Cain, 2011). Taxonomic classification systems exist within many domains of research including software engineering, database work, genealogical recording, bibliometrics, and others, but the idea is most typically associated with the classification of organisms in the biological sciences. Taxonomies and hierarchies are often visualized using tree diagrams, a visualization technique that dates back centuries and whose early works are often attempts to depict the ancient classification systems developed by philosophers such as Aristotle.

In *The Book of Trees: Visualizing Branches of Knowledge* by Lima (2014), various kinds of tree diagrams and their historical origins are depicted. Tree diagrams are not limited to the typical visual structure that actually looks like a figurative tree, and the concept of visualizing hierarchical structures has taken on more and more abstracted forms over time, each with their unique advantages. There are eleven tree diagrams described in Lima's book. Some points on each tree diagram adaptation are provided here:

Figurative Tree Diagram: Original tree diagrams that looked like literal trees. Diagrammatic terms like root node, branch link, and leaf were adapted from this original metaphor.

Vertical Tree Diagram: Common tree diagram abstracted from figurative trees. Nodes and links can incorporate a variety of visual features, symbols, or icons to convey information.

Horizontal Tree Diagram: Common tree diagram with correspondence to left-to-right reading.

Multidirectional Trees: Tree diagram with no hierarchical ordering along a particular spatial axis. Effective for very large hierarchical systems as visual space is optimally used, but ranking structure becomes unclear due to flexible layout.

Radial Tree: Tree structure ordered circularly, with the root in the center and sub-ordinate hierarchies moving towards the peripheral. Spatially efficient compared to tree diagram.

Hyperbolic Tree: Radial tree typically rendered in digital space where some links and nodes are more emphasized than others. Optimizes screen space that radial trees might waste.

Treemap: Space filling visualization depicting hierarchies using nested rectangles. Size of rectangles can correspond to quantitative attributes. Spatially efficient and legible.

Voronoi Treemap: Similar to Treemaps but use polygons rather than rectangles to present hierarchies. Size of polygons can correspond with quantitative attributes. More

spatially optimized than Treemaps and hierarchies are easier to distinguish due to non-recurring shapes.

Circular Treemap: Similar to other treemaps, hierarchies depicted as shapes within shapes, and size can correspond to data. Unlike other treemaps this type wastes visual space.

Sunburst: Treemap structure ordered circularly. Hierarchies depicted with root in center and sub-ordinate hierarchies moving towards the periphery. Size of shapes can correspond to data. Spatially efficient but distinctions between layers can be hard to perceive.

Icicle Tree: Combination between conventional Tree Diagram and Treemap. Organized top-to-bottom or left-to-right, hierarchy is shown as adjacent rectangles to show rank. Size of shapes can correspond to data. Not as spatially efficient as conventional Treemap.

In addition to the challenge of visualizing hierarchical structures, our visualization problem also involves revealing attributes associated with individual software, which in the current project context are found as the most subordinate leaf nodes. Visualizations showing software and associated attributes can be found in software engineering research. One such example comes from Grimstead et al (2005) where they create a list of categorized software systems and score them based on their scalability. In the table they present, these software systems are each associated with the presence or absence of some attributes. To summarize the data, the researchers made numerous attribute values and averaged them, and then visualized the scores along the axes of a polar plot to create shapes for each software group. Note that leaf-level taxonomical information is lost in the averaged polar plot.

Another example of taxonomy in the area of software development comes from Kucher and Kerren (2015) where they visualized a taxonomic classification system for text-based visualizations. Their taxonomy is an attempt to understand the inventory of text-based visualization that is a result of social media creating an explosion of data to mine. In their paper these authors offer a taxonomy of text-based visualization software as well as a browser-based visual survey of text-based visualizations that they have identified (http://textvis.lnu.se/).

For our taxonomy and attribute visualization to be comprehensive, we must be able to show connections at the leaf-level that reveal individual software to be compared. Types of node-link visualizations showing leaf-level connections include arc-diagrams and bi-partite diagrams. Arc-diagrams draw link connections between nodes that exist along a single axis (Wattenberg, 2002) while bi-partite diagrams show linked associations between two sets of nodes representing distinct classes (Wang et al, 2015). An example of an arc-diagram depicting relations between software task types is a visualization created by Autodesk in 2010 depicting which application tasks are most used in their 3D modelling software 3Ds Max. In contrast, a bipartite graph from a neuroscience article showed which of two sets of brain regions are connected (Bohland et al, 2009).

While not comprehensive, the above brief literature review offers a view of the landscape for hierarchical and software taxonomic visualization. Interestingly, we did not find an identical example of software taxonomy visualization that solves the exact same problem as our current project mandate. While the taxonomic systems visualized in the literature showed

categorization schemes for software, we have an entire database of software that must be organized into a hierarchy and sorted based on external attribute values.

This database consists of a broad range of software technologies that were chosen at the inception of the project as appropriate for use in transportation planning and analysis. We concentrated on tools that incorporated elements of analysis and visualization, and spanned two and three-dimensional approaches to visualization. We were guided by our comparative evaluation of two applications that were under development as part of the iCity project: Betaville and StoryFacets (Dunne et al, 2016). The Dunne et al. research considered applications that we built and maintained for general public use, with these two applications acting as different examples of analysis capability and visual representation. We recognize that the list of software that we analyzed for Compara are a limited sample constrained by context and timing. We did not capture the constant infusion of new software and upgrades of existing software that occurs in urban informatics.

In the next section, we outline the design process and results of our prototyping efforts.

Methods – Design Process

To solve the visualization problem we undertook design sketching and rapid-prototyping. For our visualization to be useful, we aimed to visualize as much of the spreadsheet information as possible without making the visualization confusing or difficult to understand. We started our design process focused mostly on the hierarchical structure of the software taxonomy and later integrated attribute visualizations. Below images that demonstrate our design sketching and thinking process. Some are hand-sketched and others created with digital tools. A description of each of the sketches is provided below each figure.

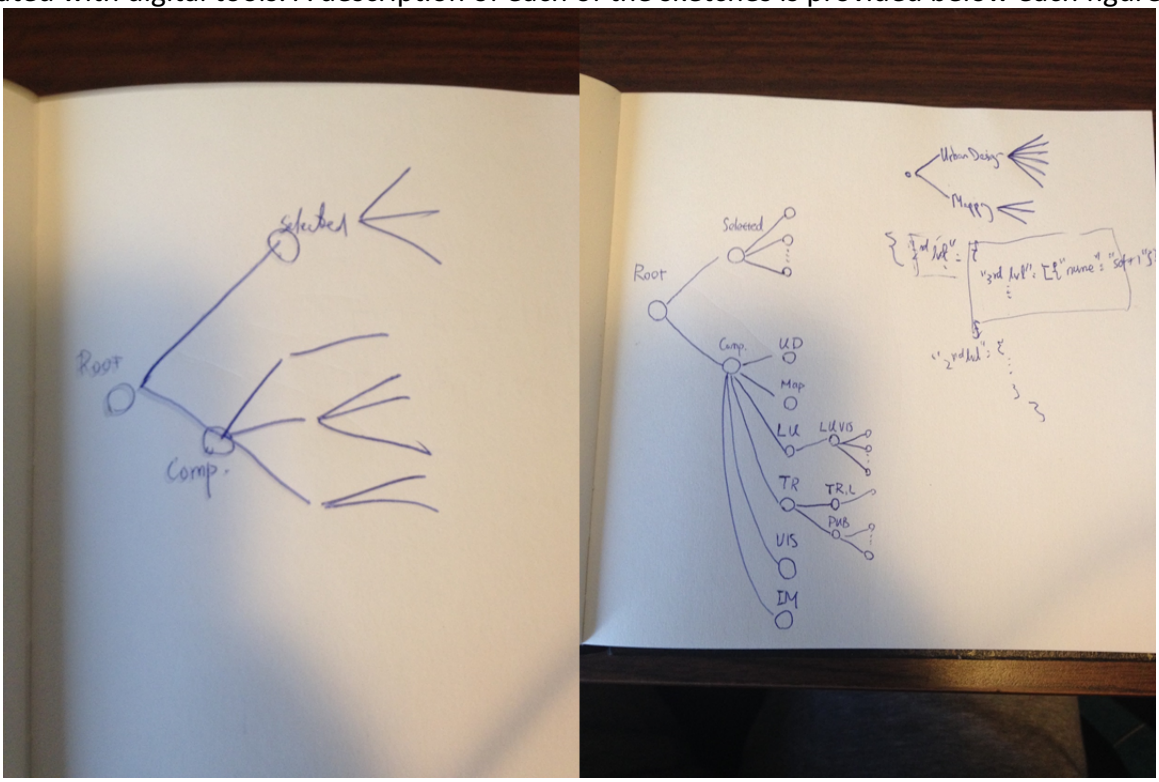

Figure 2: Early sketching ideas for the software taxonomy. The first instinct was to begin thinking about standard tree diagrams.

We created a tree diagram as a starting point – Figure 3. The root node represents the entire dataset, which branches out into software categories. The final sub-ordinate level, the leaf level, consists of each software system. This diagram is an example of a multidirectional tree diagram and was manually constructed using Simplemind, a mind-mapping application. This first step was promising as it represents our first digital representation. This visualization however was hand-made and not generated by inputting data and running an algorithm, and therefore was not a suitable finished solution for this project.

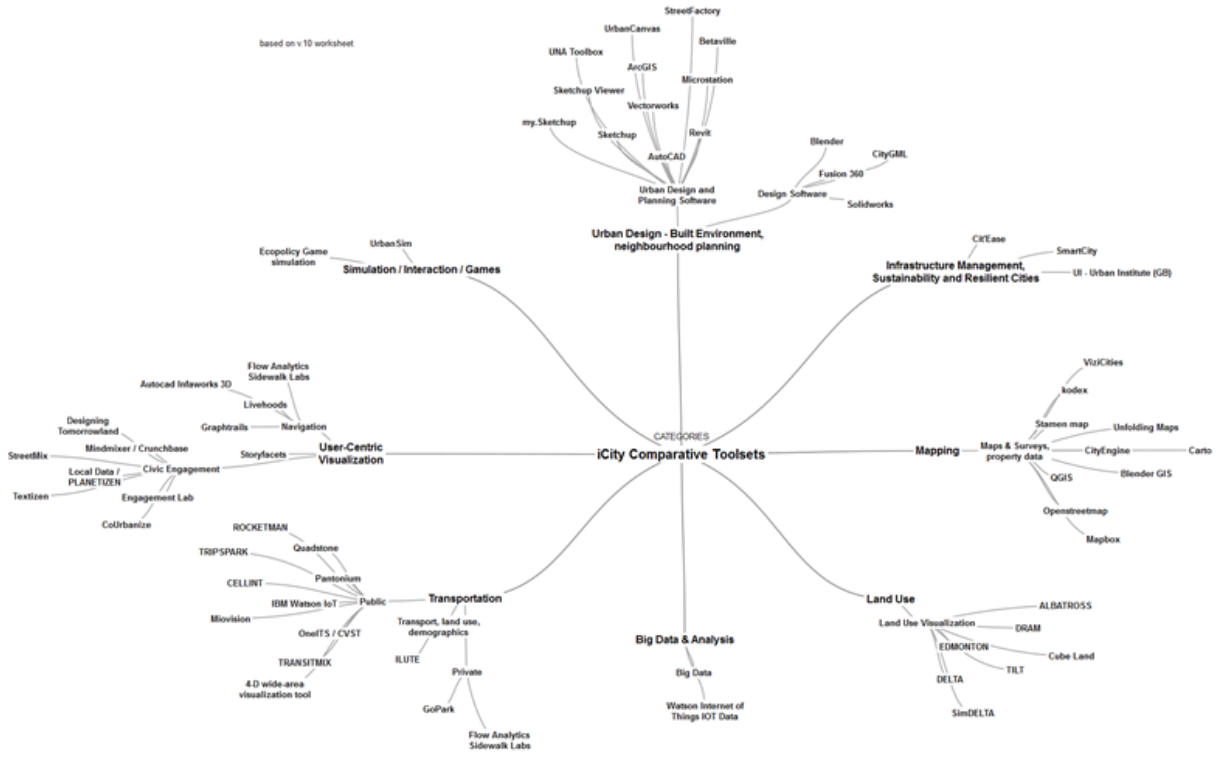

Figure 3: Our first tree diagram showing the software taxonomy categorized in a hierarchical structure.

In parallel to the taxonomy research (Bowes et al 2018), the software examples were organized into eight categories. These groupings represented tasks that professionals in urban informatics performed. We experimented with a series of different ideas. We first planned to connect the leaves of the tree diagram to show relationships between software through arcs. We then created a sunburst-like visualization, with a chord diagram in the center to show relationships. We later tried force-diagrams, a type of multidirectional tree diagram using web-based spatial auto-organization based on weighted relationships. Finally, we attempted treemaps with inner-connections between leaves, and the same using circular treemaps. In all

cases, we tried to represent a hierarchy, and then introducing new lines or visual variables. None of these represent a comprehensive solution, but this was an important brainstorming session when considering various options.

We then revisited the original tree diagram but added external nodes as illustrated in Figure 4. Figure 3 can be seen in the center of the diagram, while the clusters are the periphery represent the external attributes. Connections from the taxonomy leaves to the attribute nodes visualizes which software is associated with which of the attributes. This visualization is a precursor to our final project solution, but reveals some important lessons and limitations. What this indicates is that while this complex node-link setup does in fact show all the information simultaneously, it remains rather confusing and difficult to understand. Tracing the connections from attributes to software taxonomy is difficult to follow, and the image is too busy to easily follow and see the intricate relationships. This visualization was also made manually using Simplemind.

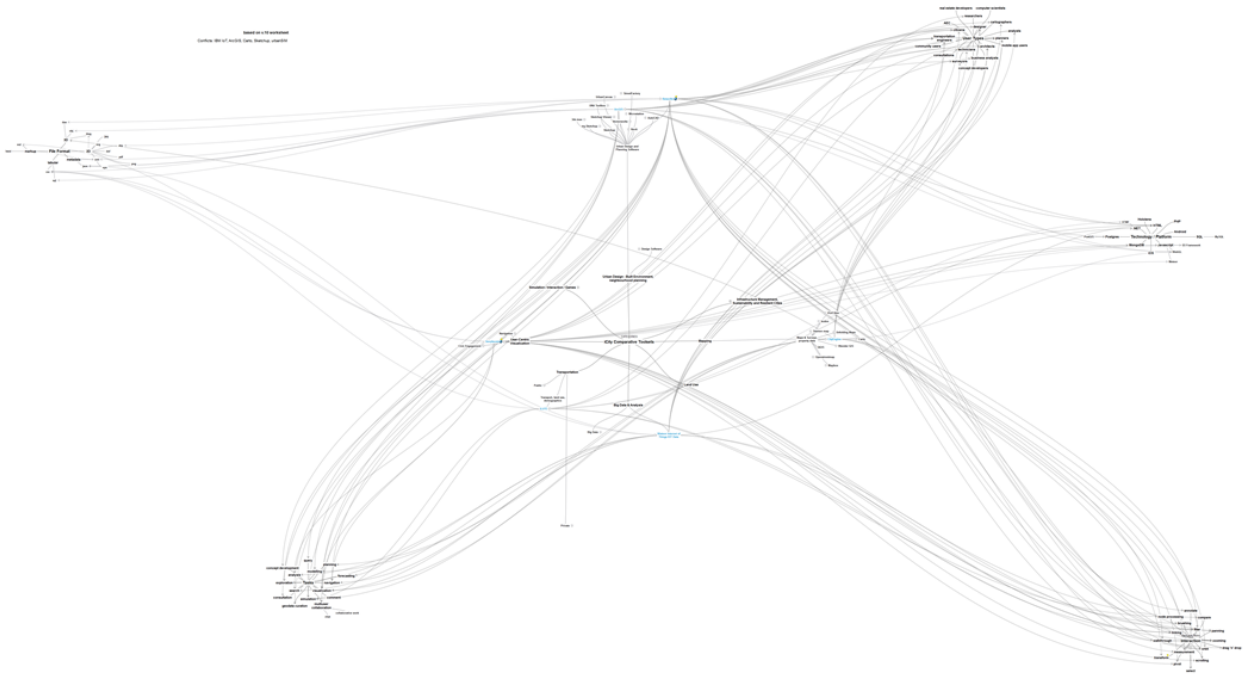

Figure 4: Revisiting our initial tree diagram from Figure 3, but now adding external nodes to represent the attribute information.

We experimented with the idea of connecting multiple tree diagrams in an ordered fashion, as in Figure 5, but it became clear that this was interesting but unnecessary, and that the concept of a bipartite graph (mentioned in the literature review) would be sufficient for matching software with attributes. At this level of design thinking we became confident that our visualization concept of combining tree diagram with a bipartite graph could actually work to represent both the taxonomy and the associated attributes. From here we felt comfortable to begin prototyping using more committed and advanced methods, as will be shown below.

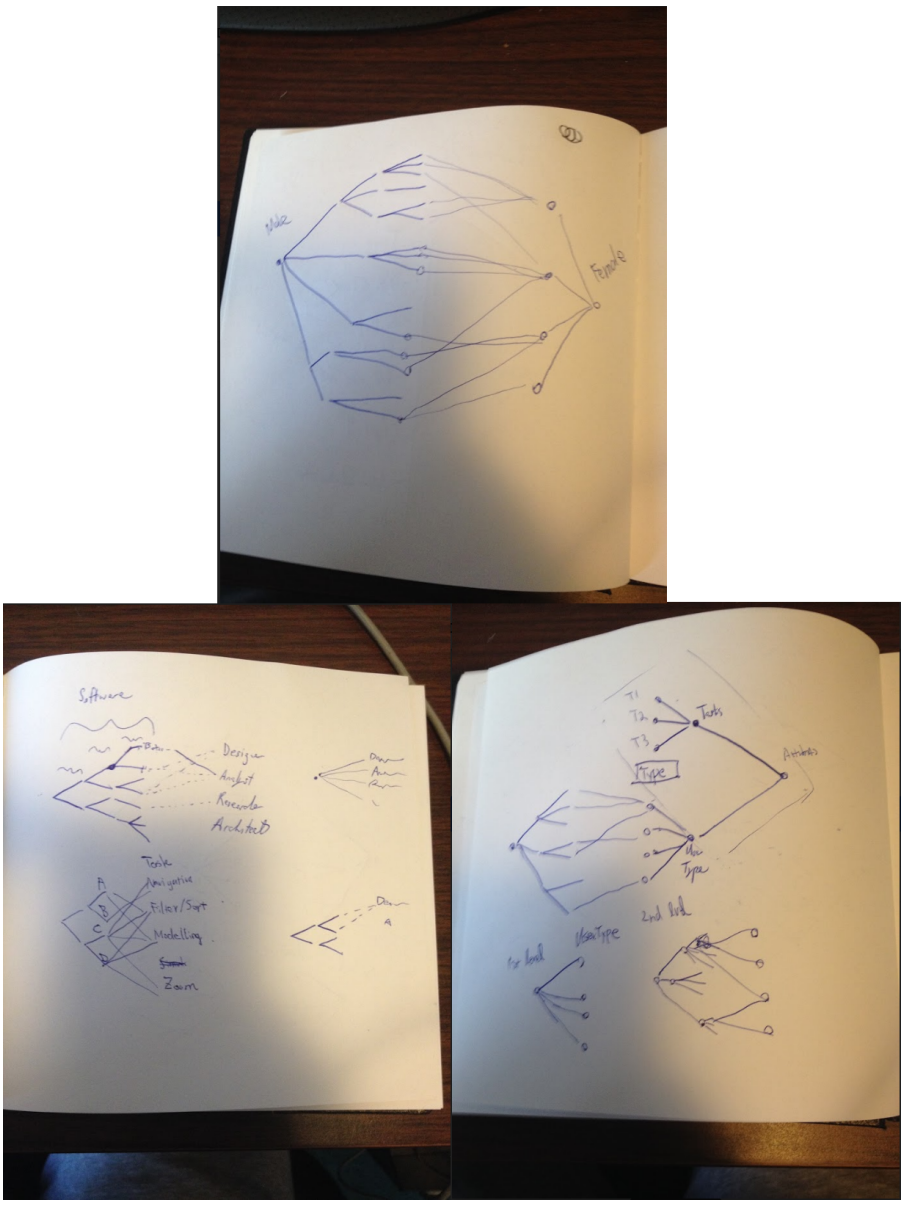

Figure 5: Design sketches showing our conceptualization for connecting an organized horizontal tree diagram representing the taxonomy, with an external set of nodes representing the external attributes.

Results - Design and Implementation of the Tree Diagram / Bipartite Graph

The final result of our design process and implementation is an interactive web-based tree-diagram/bipartite graph. This section will describe the functionality and characteristics of the final prototype, as well as discuss the design process for the final iterations. Figure 6 shows the final iteration of the tree-diagram/bipartite graph that shows the entire software taxonomy as well as associated attributes for each leaf-level software platform.

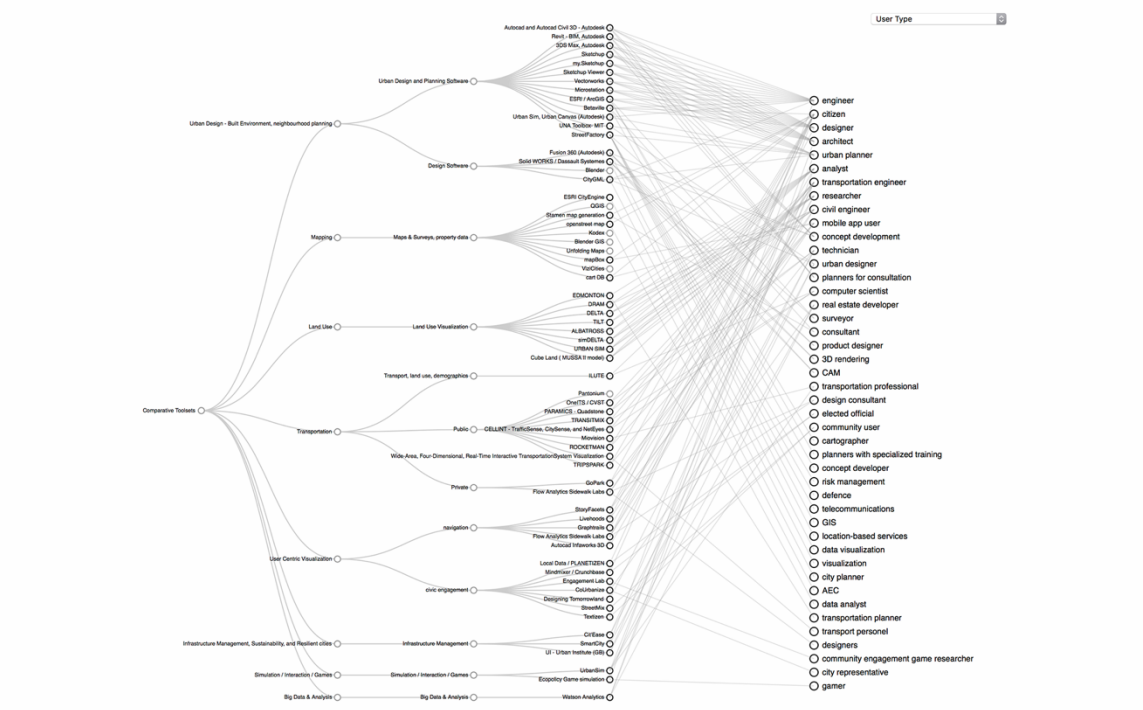

Figure 6: Final tree-diagram/bipartite graph visualization of the software taxonomy and attributes dataset.

The horizontal tree-diagram (left) shows the hierarchical categorization structure of the individual software platforms that are found as the sub-ordinate leaf level of the tree. The bipartite graph is composed of the software leaf-level from the tree-diagram, combined with the set of nodes and links on the right, which represent connections to qualitative attribute values for the category of "user-type". Effectively, this visualization shows the entire taxonomy of software and the types of users (e.g., designer, planner, citizen, etc.) most likely to benefit from using each software. At the top-right, one can see a drop-down menu that allows the user to switch attributes from "user-type" to "task type" and vice-versa. Interactive elements were added to the visualization in order to enhance its usefulness in terms of highlighting software/attribute associations for analysis, switching between sets of attributes linked to the tree-diagram, and increasing readability. Five interactive elements include link highlights (Figure 7), node highlights (Figure 8), node selection (Figure 9), attribute switch (Figure 10) and zoom.

(1) Link Highlight

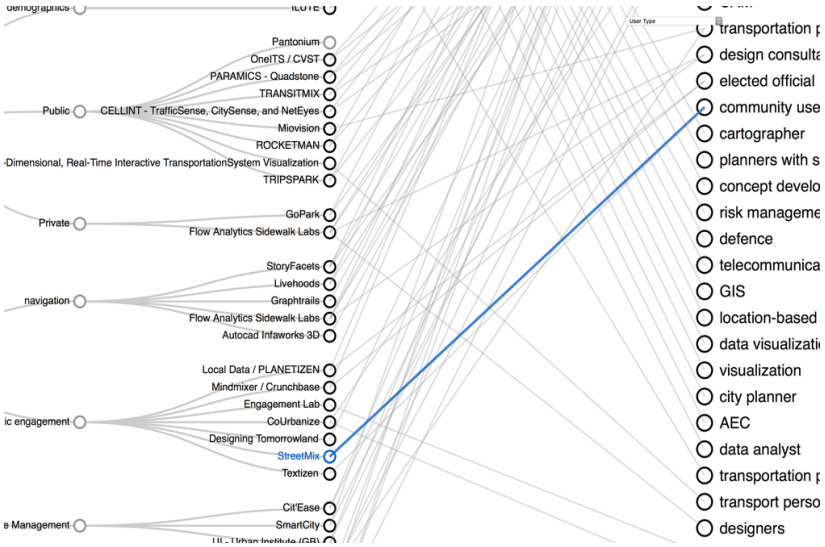

Figure 7: When the mouse hovers over a node or link between software and attribute, the node-link connection is highlighted in blue. If a node has multiple links, all attached links are highlighted.

(2) Node Highlight

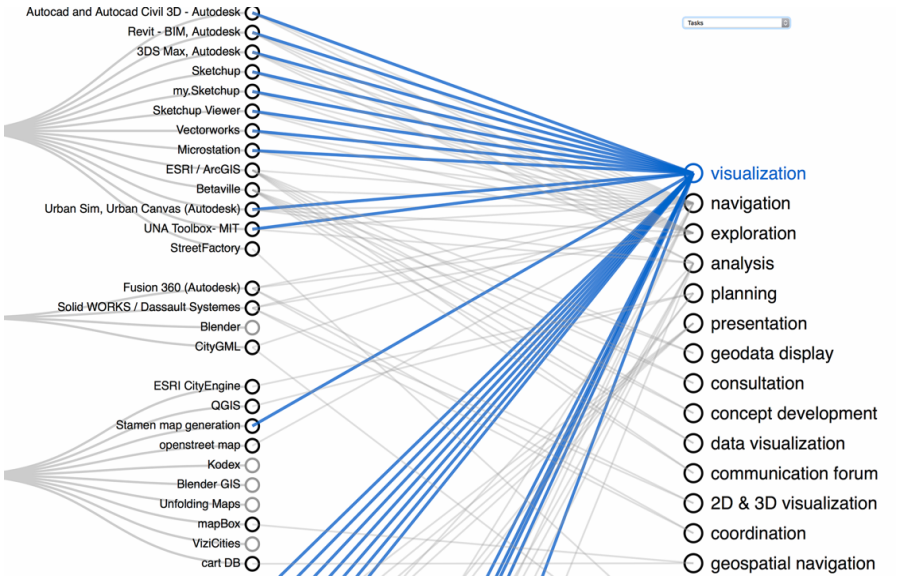

Figure 8: When the mouse hovers over a node (either software or attribute), the connections to that node are highlighted in blue. This acts to make more clear which attribute features are related to which software platforms, and vice-versa. The bipartite graph with its grey links is difficult to interpret on its own and this was added to help with exploring relationships.

(3) Node Selection

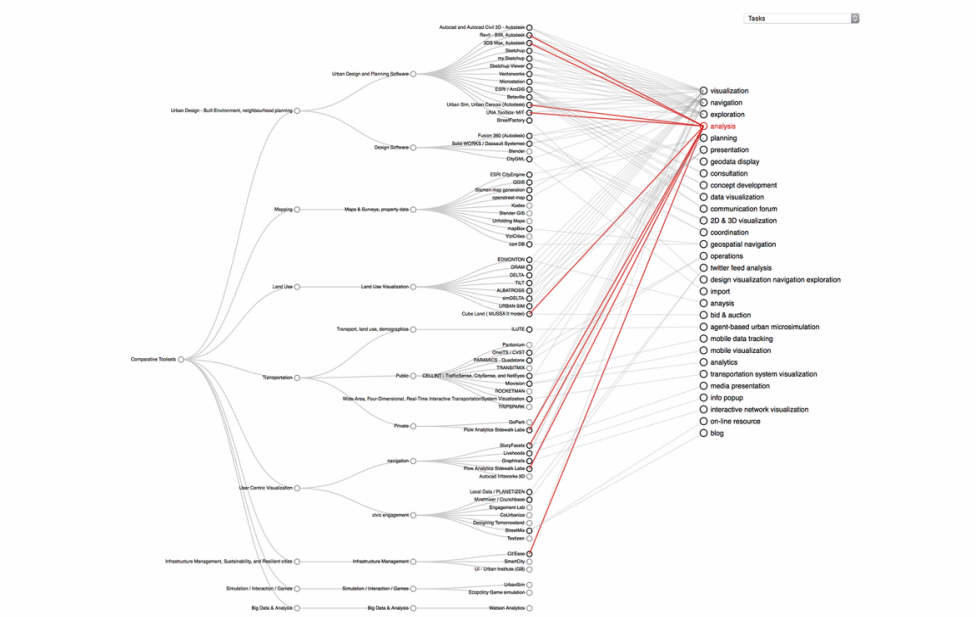

Figure 9: When users mouse-click on a node, the connections to that node are highlighted and locked until further clicks. This allows users to compare one set of selected connections to a second set that is highlighted by mouse-hover.

(4) Attribute Switch

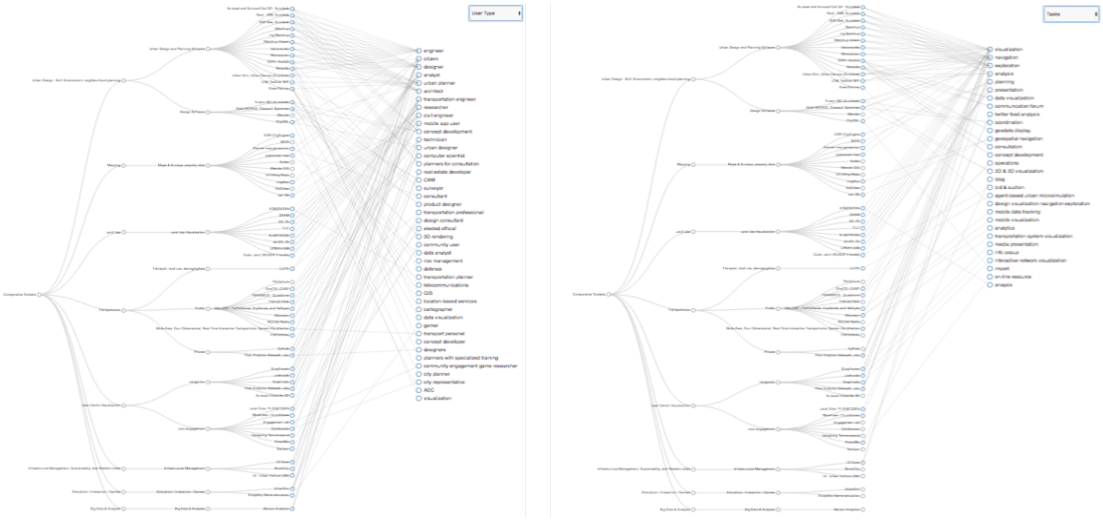

Figure 10: Using the drop-down menu at the top-right of the visualization users can switch between attributes and their value sets on the right side of the bipartite graph. The two attribute sets include “user types” (left) and “tasks” (right).

To increase the legibility of our graph, users can zoom close up to the visualization to read text or focus on specific visual regions. The zoom centers on the current mouse location.

Implementation

The implementation of our tree-diagram/bipartite graph is described below. To implement the design for the tree-diagram/bipartite graph, we decided to use the web-browser visualization technology D3. We chose D3 as there are many visualization techniques previously implemented in D3. D3 is a robust Javascript library with rich development resources and extensive community support and it is open source. It effectively supports interactivity. It is easy to integrate D3 into any web application framework. There will thus be less required effort in producing a web-based dashboard application that could be useful for presenting to OCAD U researchers or stakeholders.

We wrote a script that produces the visualization in a web-browser. The script parses and models the data read from a JSON file and renders a hybrid visualization. The data processing is shown in Figure 11 System diagram.

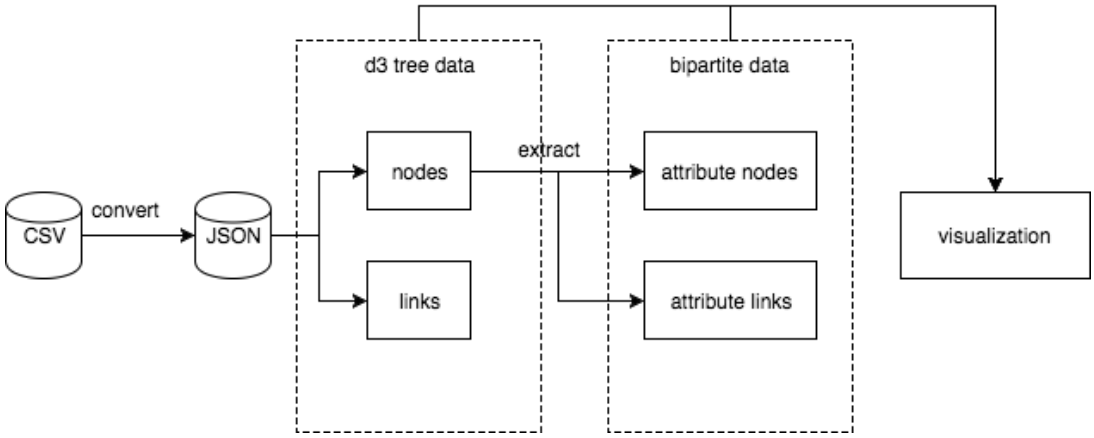

Figure 11: System diagram. First, a CSV file is converted to a JSON file. Next, the data gets parsed and modelled from a JSON file into d3-compatible data structure. Finally, the visualization is rendered.

Since D3 works best with standard file formats such as CSV and JSON, we manually extracted and formatted the data from the raw spreadsheet file to a CSV file. Then we converted the CSV file to a JSON file using an external web service with a custom template.

We explain the algorithm to build the tree and bi-partite diagram in terms of node and link construction. For the nodes and links of the tree diagram, D3 has built-in support for generating a tree layout object from a provided JSON data file. The D3 code for creating collapsible tree diagrams can be found at https://bl.ocks.org/mbostock/4339083. For the styling of the leaf nodes, nodes with connections or links (as part of the bi-partite diagram) are in blue and those without connections are in grey.

For the bi-partite diagram, since the raw data itself does not contain explicit relationships, we had to do some text mining to extract and construct relationships between software leaves and its attributes. There are currently two attributes used (i.e., “user type” and “user task”) from the larger dataset spreadsheet which contains others. For example, given an attribute called “User Type”, we construct a list of nodes of distinct user types from the raw data. We draw a link between a software node and a user type node if the software supports that user type. We also sort the list of attribute nodes by their number of connections from

most to least in order to reduce overlap between links, thereby making the visualization more legible. It should be noted that the bipartite aspect of our visualization was custom coded in d3 by the authors of this project.

In terms of interactivity, we implemented pan/zoom and node/link highlighting and selection. Since the visualization is scaled to fit the browser window, nodes and texts are scaled down and may not be legible. With panning and zooming, users are able to freely adjust the area of focus in the visualization. The compound visualization consists of multiple levels and numerous nodes and edges, which may take higher cognitive loads to trace the origin of one or more links. We implemented node and link highlighting to ease that problem. Currently, users can highlight a single link between a software node and an attribute node by hovering the mouse cursor over the link. Alternatively, they can hover on either side of the nodes to highlight a node label and its links if there is any. Clicking the mouse will let the highlight persist, but only one node or link can be highlighted at a time.

Future Improvements for Prototype

While our visualization results succeeded in fulfilling the initial criteria for this project, our visualization could be improved and developed further. Below is a list of suggested immediate improvements that can be made:

- Nodes in the hierarchy should be collapsible, so that users can observe connections made between the hierarchy and attributes at different levels of the hierarchy. This would allow the comparison of aggregated data rather than only individual software platform data points.
- To optimize the visual space and reduce clutter, the link nodes could be made into curved lines rather than straight lines.
- The node links could be optimized to further reduce edge crossings, thus reducing visual clutter. This process would be algorithmic, and automatically optimize for every new set of attributes or changes to the visualization.
- Highlighting parent nodes in the tree diagram should highlight its ancestor nodes and links as well as its children nodes and links in the tree diagram and the bi-partite graph. This would offer a more comprehensive exploration by the user.

Integrating the Visualization into a Comprehensive Dashboard Application

Here we propose a design for taking our visualization a step further and integrating it into a web-based dashboard application that offers information over and beyond what is already available through the tree-diagram bi-partite graph. We have named this proposed design a Project Compara. Figure 12 shows a visual mockup of the dashboard application layout.

To visualize the data as a tree, a web-based dashboard design is created to house the visual and have other support data views to complement it. Based on our research of visualization methods, the force-directed layout as well as treemap reflect the ability to place focus on relationships and hierarchy, respectively.

The dashboard design is created with four functional sections for analyzing the data of the toolsets. The main section is designed to have the main visualization type as a tree, with the root titled as the dashboard title itself: Compara. It expands out into the first level of headings, which expands out into another group of subheadings of the software tools. Changing this tree structure would be based on the JSON data file that is loaded into the dashboard.

The next section consists of a force-directed layout that provides the user the ability to seek direct relationships between software packages. The third section acts as a middle ground, providing general information about the selected tool, including title, product description, the highlighting of the user types and tasks associated. The last section is a treemap visualization that places focus on hierarchy, from main category of software down to the software itself. Additional items considered for the treemap include a list of keywords that contribute to the rationale and description of the categories. This places emphasis on hierarchy allowing the user to have a clear understanding of how software selected in the tree relates to the subject matter of focus, intended by the manufacturer of the tool.

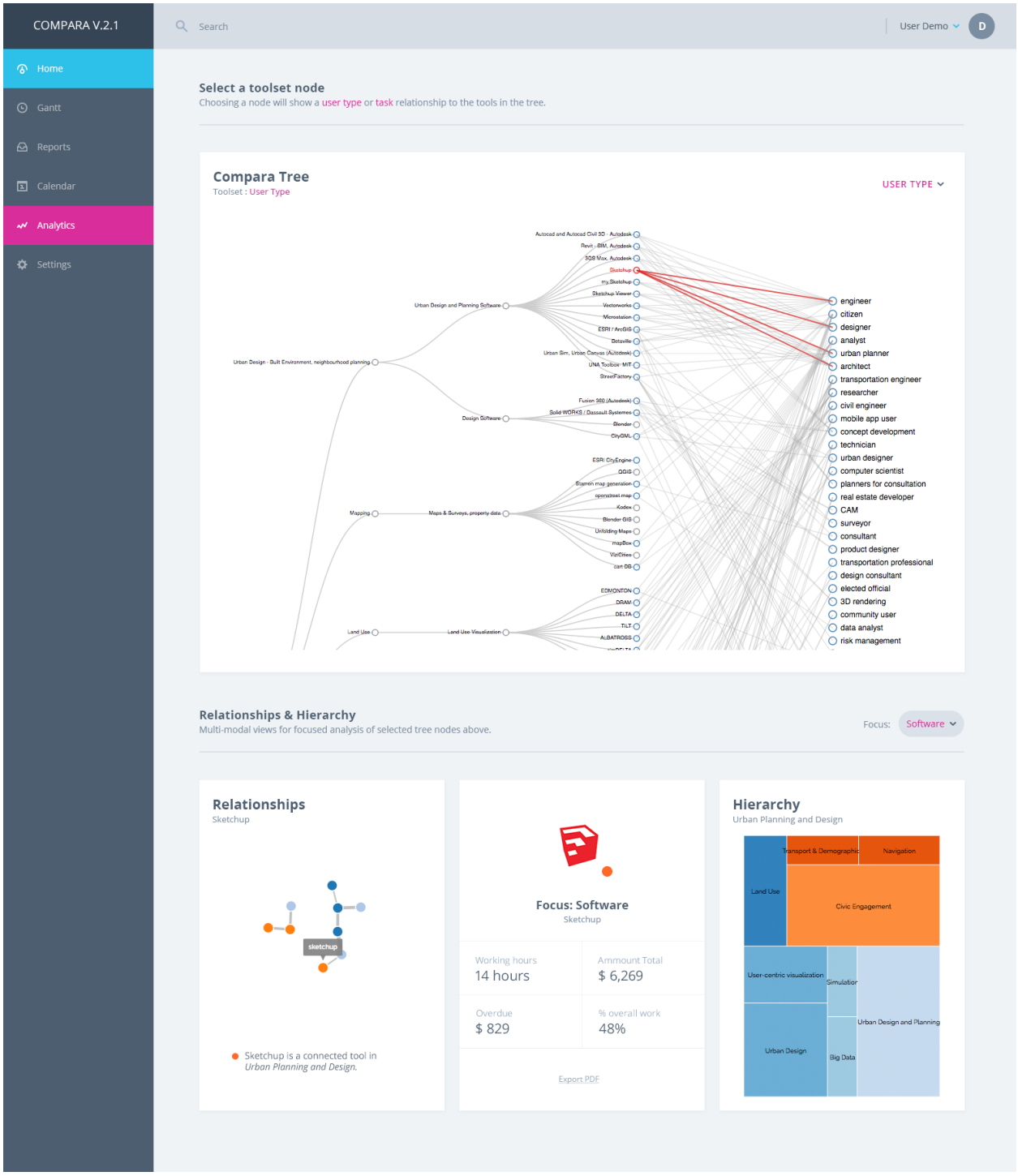

Figure 12: Project Compara is a dashboard design created to house the visualization, as well as integrate other support data views to complement it.

Based on our research of visualization methods, the force-directed layout (bottom-left) as well as treemap reflect the ability to place focus on relationships and hierarchy, respectively. Extra D3 visualization methods can be daisy chained from the original data set, and further

qualitative information can be embedded to offer extra information regarding software platforms or other features (bottom-middle).

Conclusions

To conclude we will review the initial criteria of this project including the project purpose and the objectives and questions to evaluate the success of our visualization. Finally, we will discuss potential future improvements as well as possible applications for our visualization project.

We believe our visualization is a viable solution to the stated purpose of this project. We visualized the entire software taxonomy, making clear its hierarchical structure in a way that is organized and understandable using a horizontal tree-diagram. We implemented a bipartite graph to successfully visualize how attributes are connected to software platforms, even when some software or platforms have multiple relationships. We achieved one of the central uses for this visualization project, which was to help users identify which software are associated with specific user-types. We also created a way to discover relationships between different sets of attributes and software platforms using the drop-down menu. We believe this visualization technique can be generalized to other domains. This visualization can definitely be scaled to datasets of different sizes to show different hierarchical structures and associations, but comparing qualitative attributes becomes difficult when the structure becomes too large.

Based on our initial objectives and questions, we believe our visualization has appropriate answers for each. In order to visualize a hierarchical structure and associated external attributes, we implemented a tree-diagram/bipartite graph. From our own research, we cannot find an example of this kind of visualization being used in any similar context. It is possible that we have created an innovative visualization approach. We converted a structured spreadsheet into a visualization that makes the hierarchical structure much clearer than a traditional tabular format. We believe that our visualization has translated an indexical structure into something that is both aesthetic and engaging, incorporating smooth D3 vector visuals with exploratory interactive features. Finally, we believe this visualization is most effective when it supports user interactions, which allow users to solve the perceptual problems that arise when dealing with an intricate and dense visual structure as this one.

REFERENCES

Bohland, J. W., Bokil, H., Allen, C. B., & Mitra, P. P. (2009). The brain atlas concordance problem: quantitative comparison of anatomical parcellations. *PloS one*, *4*(9), e7200.

Bowes J. et al. (2018). User-Centered Taxonomy for Urban Transportation Applications. In: Nah FH., Xiao B. (eds) HCI in Business, Government, and Organizations. HCIBGO 2018. Lecture Notes in Computer Science, vol 10923. Springer, Cham

Cain, A.J. (2011). Taxonomy. In Encyclopaedia Britannica Online. Retrieved from https://www.britannica.com/science/taxonomy

Dunne C., Skelton C., Diamond S., Meirelles I., Martino M. (2016) Quantitative, Qualitative, and Historical Urban Data Visualization Tools for Professionals and Stakeholders. In: Streitz N., Markopoulos P. (eds) Distributed, Ambient and Pervasive Interactions. DAPI 2016. Lecture Notes in Computer Science, vol 9749. Springer, Cham

Grimstead, I. J., Walker, D. W., & Avis, N. J. (2005, October). Collaborative visualization: A review and taxonomy. In *Distributed Simulation and Real-Time Applications, 2005. DS-RT 2005 Proceedings. Ninth IEEE International Symposium on* (pp. 61-69). IEEE.

Kucher, K., & Kerren, A. (2015, April). Text visualization techniques: Taxonomy, visual survey, and community insights. In Visualization Symposium (PacificVis), 2015 IEEE Pacific (pp. 117-121). IEEE.

Lima, M. (2014). *The book of trees: visualizing branches of knowledge*. S. E. Stemen (Ed.). Princeton Architectural Press.

Wang, L., Wu, H., Wang, W., & Chen, K. C. (2015). Socially enabled wireless networks: resource allocation via bipartite graph matching. *IEEE Communications Magazine*, *53*(10), 128-135.

Wattenberg, M. (2002). Arc diagrams: Visualizing structure in strings. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on* (pp. 110-116). IEEE.

SOFTWARE SURVEYED

Autocad
https://www.autodesk.com/products/autocad/overview

Revit
https://www.autodesk.com/products/revit/overview

3ds max
https://www.autodesk.ca/en/products/3ds-max/overview

Sketchup
https://www.sketchup.com/products/sketchup-pro

My.Sketchup
https://www.sketchup.com/products/sketchup-free

Sketchup Viewer
https://www.sketchup.com/products/sketchup-viewer

Vectorworks
https://www.vectorworks.net

Microstation
https://www.bentley.com/en/products/brands/microstation

ArcGIS
https://esri.ca/en/products/arcgis-pro

Betaville
https://github.com/Betaville

UrbanSim
http://www.urbansim.com

UNA Toolkit
http://cityform.mit.edu/projects/urban-network-analysis.html

StreetFactory

Fusion 360
https://www.autodesk.com/products/fusion-360/overview

SolidWorks
https://www.solidworks.com

Blender
https://www.blender.org

CityGML
https://www.citygml.org

CityEngine
https://www.esri.com/en-us/arcgis/products/esri-cityengine/overview

QGIS
https://www.qgis.org/en/site/

Stamen Map
http://maps.stamen.com

Openstreetmap
https://www.openstreetmap.org

Kodek

BlenderGIS
https://github.com/domlysz/BlenderGIS

Unfolding Mpas
http://unfoldingmaps.org

Mapbox
https://www.mapbox.com

Vizicities
https://github.com/UDST/vizicities

CartoDB
https://carto.com

EDMONTON
DRAM
DELTA
TILT
ALBATROSS
simDELTA

Cube Land
http://www.citilabs.com/software/cube/cube-land/

ILUTE
http://uttri.utoronto.ca/research/projects/icity/icity-orf-research-day-2018/ilute-integrated-land-use-transportation-and-environment-model-reboot/

Pantonium
https://pantonium.com

OneITS / CVST
http://cvstproject.com

PARAMICS
http://www.paramics-online.com

TRANSITMIX
https://www.remix.com

CELLINT

http://www.cellint.com

Miovision
https://miovision.com

ROCKETMAN
http://www.rocketmanapp.com

TRIPSPARK
https://www.tripspark.com

GoPark
https://www.go-parking.com

Flow Analytics / Coord
https://coord.co

StoryFacets

Livehoods
http://livehoods.org

Graphtrail
https://dl.acm.org/citation.cfm?id=2208293

Infraworks 3D
https://www.autodesk.com/products/infraworks/overview

Crunchbase
https://about.crunchbase.com

Engagement Lab

CoUrbanize
https://courbanize.com

Streetmix
https://streetmix.net/-/754979

Textizen
https://www.textizen.com

Citi'Ease

Ecopolicy Game Simulation
http://www.frederic-vester.de/eng/ecopolicy/

Watson Analytics
https://www.ibm.com/watson-analytics