Providing Form to Function: Exploring Code Through Visualization

xCoAx 2025
13th Conference on Computation, Communication, Aesthetics & X
Dundee, UK
2025.xCoAx.org

As digital spaces increasingly shape our daily lives, there's a growing need to better comprehend how the software we use functions. Relying on digital technology without understanding its workings diminishes user agency, and a lack of understanding reduces its accountability. However, by learning about how software operates and its constructed, we can reclaim agency in the digital technologies we use. Throughout this paper, we explore the relationship between digital literacy, computational thinking, and agency, and argue that information visualization can serve as a transparent method to make abstract code more accessible. *Exploring Hidden Worlds* is an exploratory tool that merges information visualization and programming language analysis to visualize code in three-dimensional space. This interactive approach empowers users by enabling them to explore code at their own pace and uncover the underlying structure of software development. The project draws from computer science, information design, communication design, and user experience design to envision a novel way to interact with the intangible nature of code.

1. Introduction

In our daily lives, we interact with numerous digital interfaces and experiences, making each scroll or button press feel insignificant in the grand scheme of things. While interacting with a single social media post or writing an AI prompt may seem inconsequential, collective interactions can lead to significant effects on our behavior. However, users often have limited control over changes to frequently used interfaces as they are updated and streamlined (Chun 2016, 2), particularly when it comes to aggregated personal data. The fields of user experience and communication design are grappling with issues related to "the digital" (Stambler, Veeramoothoo, and Davis 2024), and to better engage with and retain agency in increasingly digital spaces, there is a need to enhance our understanding of digital environments and how they are constructed. If we were aware of the ways our data is collected and the attention it draws from us, our relationship with digital technology would likely be different. Following the argument that intelligence is a form of agency (Satyanarayan and Jones 2024), by understanding how software works and how it is constructed, we can regain some control over the digital

Matthew Blanco
Northeastern University, Boston, United States
Blanco.m@northeastern.edu

Todd Linkner
Northeastern University, Boston, United States
t.linkner@northeastern.edu

Keywords Human-Computer Interaction, Information Visualization, Knowledge Transfer, Computational Thinking, Digital Literacy, Visualizing Complexity

DOI 10.34626/2025_xcoax_012

technologies we use. Though there is a push to teach programming skills and computational thinking through abstraction of algorithms, it is equally important to consider the impacts code has both within the function it makes and the impacts on users of software.

Meanwhile, digital data serves as a valuable resource for informing public policy, scientific discoveries, business strategies, and our personal lives (Heer and Shneiderman 2012). It becomes particularly advantageous when combined with information visualization's ability to simplify complex information (Abbate and Marino 2022). Information visualizations act as a medium for knowledge transfer, making them well-suited for teaching people about how software is built. In conjunction with information visualization, interaction design plays a significant role in enhancing cognitive abilities within a visualization system (Dimara and Perin 2020). Interactive visualizations also encourage users to explore and comprehend data at their own pace, providing a deeper understanding of causes and effects that might otherwise be challenging to discern (Abbate and Marino 2022). Programmers have recognized the benefits of visualizing code, leading to the development of various tools that utilize visualization as a method for code comprehension (Hawes, Marshall, and Anslow 2015; Balzer, Deussen, and Lewerentz 2005; Wettel and Lanza 2007). These tools were primarily designed for engineers and developers, and outside of classroom settings, there are limited efforts to educate non-technical audiences about the ways software works.

Throughout this paper, we will begin by reviewing existing literature on user agency within digital technology, its relationship to digital literacy, and computational thinking. Following, we present *Exploring Hidden Worlds*, an exploratory tool that utilizes information visualization and programming language analysis to visualize code with a three-dimensional space. We argue that information visualization can enhance user agency by revealing how software is constructed, and in turn be used as a method for computational thinking. *Exploring Hidden Worlds* draws from various disciplines, including computer science, information design, communication design, and user experience design, to envision and explore a novel approach to interacting with the intangible nature of code.

2. The Need to Understand Code

As a collective society, our inability to understand software and its impacts separates our agency from the responsibility of technology companies. The separation allows software to act independently and

with less accountability to its users as it marches forward towards its own goals. In addition, with the development of generative AI tools, software is taking a larger role in thinking while having little accountability to the users it serves. Satyanarayan and Jones suggest that:

> If agency is the capacity to act as an agent, accountability is the capacity to give an account of why one acts in the way one does—an explanation of how an act meaningfully advances or addresses a goal—and hence to be held accountable by oneself and others. (Satyanarayan and Jones 2024)

We have become so used to the offering of technology and new software, embedding it into our day-to-day lives that the ways our experiences are shaped by it have become invisible and deceptive (Stambler, Veeramoothoo, and Davis 2024). The goal to improve our understanding of digital technology lies in the fields of computational thinking and digital literacy. Both are related to one another, and though definitions differ, digital literacy is thought to involve skills to understand and use information from different digital sources to solve problems related to access and selection of information. Whereas computational thinking consists of problem-solving and understanding human behavior through processes of abstraction, creativity, mathematical skills, and innovation (Stambler, Veeramoothoo, and Davis 2024; George-Reyes, Rocha Estrada, and Glasserman-Morales 2021). Strengthening computational thinking skills does not require knowledge about how code is represented, or in other words:

> Computational thinking is far away from performing activities that involve writing computer code instrumentally; in other words, without mediating knowledge about the representation of the reality of the code that is being written. (George-Reyes, Rocha Estrada, and Glasserman-Morales 2021)

Therefore, understanding code enhances computational thinking, leading to increased agency when using digital devices. Filbert articulates this argument effectively, as seen in their summary:

> The ability to maintain a critical and relatively free participatory relation to increasingly computable technologies ambient and 'everywhere' must include comprehending their ontology and epistemology; design, purposes, and use; social and political functions and intentions; and be capable of altering and effecting change within them. (Filbert 2021)

Meanwhile, programming is a difficult skill to learn. Not only from the complexities and nuances of programming languages, but because of its requirement to be highly explicit (Guzdial 2008). The initiatives to improve education around programming have centered around concepts in digital literacy and computational thinking, but the fields are often ill-defined (Filbert 2021; George-Reyes, Rocha Estrada, and Glasserman-Morales 2021; Tamatea 2019). Within the effort to teach young students to code, the undertaking risks being too precise, restricting it to skills only within computer science, whereas others argue computational thinking should teach people how to exist within digital environments (Tamatea 2019). While there is disagreement over what computational thinking and digital literacy entail, researchers agree that a need exists for people to become more informed about digital technology. The ideas can be combined in a way that focuses on understanding the impacts that the structure of code has on the ways we use it. After all, if architecture informs our relationship to space (Markus 1993, 12), code informs our relationship to software.

One method to show the nuances of code is information visualization. The discipline is often used as a method to convey complex information into a simplified, easier-to-read, visual form (Masud et al. 2010). The benefits of visualizations have not gone unnoticed in computer science, and examples of visualizations for code have existed for some time (Hawes, Marshall, and Anslow 2015; Jbara and Feitelson 2014; Balzer, Deussen, and Lewerentz 2005). However, these approaches focus on visualizing code to improve code comprehension for engineers, and one drawback when it comes to code visualizations is their inability to match a developer's mental model of a codebase (Heinonen et al. 2022, 34). Simultaneously, few visualizations exist for a non-technical audience, a group that has little preconception of what code looks like but could benefit from one to improve their understanding (Tamatea 2019). Following the argument, in which a user-centered design approach to information visualization is both an outcome of the visualized "raw" data and a "transformation process" that results in the communication of new knowledge (Masud et al. 2010). Similarly, knowledge can be differentiated from information as something structured and hierarchical, resulting from a refiner's process (Cleveland 1982). Specific types of visualizations — formative visualizations — support "knowledge transfer inside cooperative work groups" to "represent workflows, processes" that can "instruct users of their role in that cooperative context" (Masud et al. 2010). The design of novel visualizations can directly aid in the transfer of new knowledge. Specifically, knowledge of how software is built. A common argument for the need for visualization is due to the increasing digitization of in-

Fig. 1. *Exploring Hidden Worlds* visualizing the open-source project Leaflet.

formation and relationships, along with the diffusion of data generated by people and organizations, each representing both a challenge and an opportunity to design tools suitable for the complex and changing environment in which we live (Abbate and Marino, 2022).

3. Giving Form to Function

Complex pieces of software can be thought about in terms of lines of code (LOC), or for exceptionally large projects, millions of lines of code (MLOC). With the frameworks that currently exist, it is common practice for engineers to only understand the subset of code they work on, and rarely have an understanding of a codebase as a whole (Hawes, Marshall, and Anslow 2015). *Exploring Hidden Worlds* attempts to create an explorable 3D visualization to give form and materiality to the intangible and difficult-to-understand infrastructure behind software, and due to the immaterial nature of code, a new visual form may be needed to effectively visualize it. The project was developed as an investigative exploration to imagine a new way of conceptualizing code. The visualization combines the hierarchical structure of a software project, from nested folder, file, and statement relationships, with referential relationships. Such as when a function invokes code from a different file. The interface is presented in a way for users to explore the novel form, and by placing the visualization in 3D space, it invites users to pan, zoom, and interact with the visualized code however they choose. Through the interface, user agency is encouraged to explore and interact with the data as they see fit, rather than enforcing a single point of truth within the visualization. The goal of the tool was to build a visualization that required little to no knowledge in programming and code to read. Following the definition of a knowledge visualization, *Exploring Hidden Worlds* aimed to help

transfer knowledge about the ways software is built from engineers to the users of the tool (Burkhard 2004).

3.1. Collecting Data

In order to build the visualization, a custom static analysis tool was written and tested with a series of open-source projects written in JavaScript. The analysis would recursively navigate through a project directory and identify JavaScript files, and proceed to identify control statements within the code (conditionals, loops, classes, interfaces, etc.) in addition to flagging statements which referenced code imported from other files. Each of the node and edge types are based on fields within a customizable schema that was used when parsing a codebase and identifying statements within code. The schema allows for a variety of programming languages to be parsed and visualized according to different syntax. The resulting analysis was then exported to a JSON object that would be independently rendered.

```
{ “language”: string
  “keywords” string[],
 “keywordsToIgonore”: string[],
 “source”: string,
 “root”: string }
```

An example of the configuration schema used by *Exploring Hidden Worlds* to analyze a codebase.

3.2. Why 3D

The visualization was developed with a 3-dimensional force-directed layout in mind to engage users by generating a more visually appealing composition, and to encourage the use of different methods of interactions to explore the network. By panning and zooming around the visualization a user becomes an active participant to figure out how different pieces of code are connected to one another, and how each node and edges informs the overall structure of a codebase. Additionally, the extra space afforded by the 3D navigational axes allows the nodes and edges within the network to be spaced out as opposed to a more traditional 2D network (Gummesson 2016). While the spatial layout introduces new challenges such as difficulties retaining orientation throughout the visualization, the obscuration of data as nodes can exist outside the viewable window, or by simply being placed further away in space. The drawbacks introduced by this novel form of visualization are diminished due to the objectives of *Exploring Hidden Worlds*. The tool was built to not act as a single source of truth

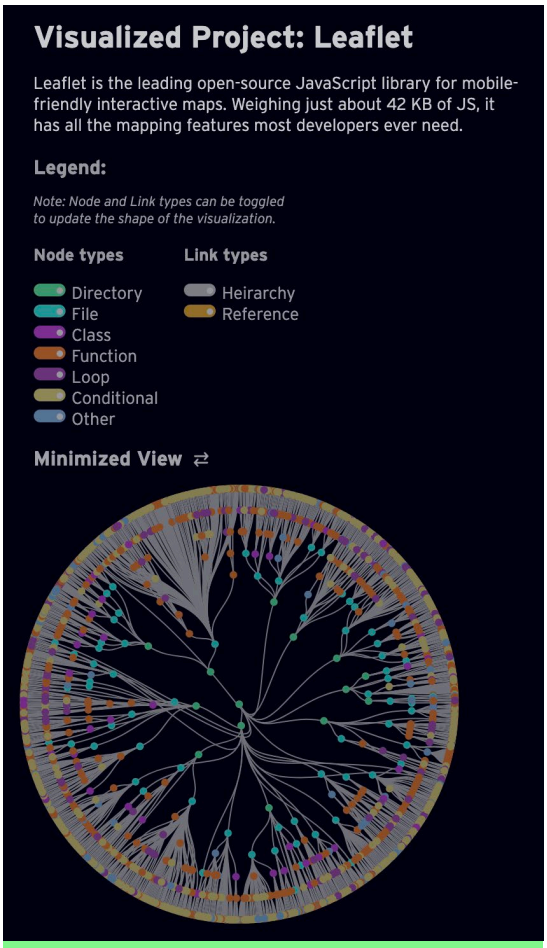

Fig. 2. Focused view of the elements within the legend.

for a user to wholly understand a codebase. Instead, much like the way an engineer approaches a codebase, a user can navigate around code using the tool to understand small portions of the visualized project. Specifically, they can pick out portions of the visualization which they find interesting and choose to dive deeper.

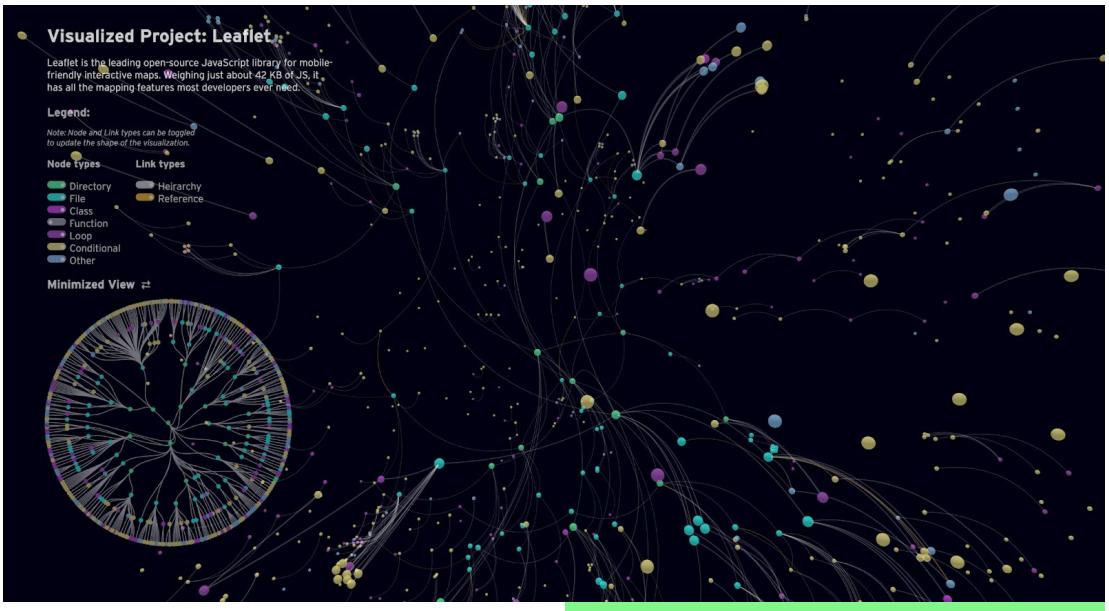

Fig. 3. Filtering out function nodes through the toggle-able legend.

3.3. About the Interface

Split into three pieces, each part of the interface utilizes concepts from information and visualization design, creating an aesthetically appealing visualization to reinforce the interactions and controls a user can do. Following Ben Shneiderman's seminal mantra "Overview first, zoom, pan, and details on demand" (Shneiderman 1996). The tool focuses on giving users a general overview of what code can look like. Only by interacting with the visualization and resulting information granularity, can a user then understand the code behind it

The legend (Figure 2) serves as the primary guide for users to understand the visualization along with: teaching a user how to read the visualization, and act as a method to filter the data and view the resulting modified data. Each type of node and edge that exists within the visualization is also toggle-able and dynamically updates the shape of the visualization when interacted with.

After each filter the 3D network is re-rendered, and the new form allows users to focus on specific pieces of data while reducing the complexity of the visualization in both the 3D network and 2D dendrogram (Figure 3 and Figure 4).

Beneath the legend sits a minimized view of the visualized code in the form of a radial dendrogram. The secondary visualization exists to showcase the structure of the code in a more familiar 2D representa-

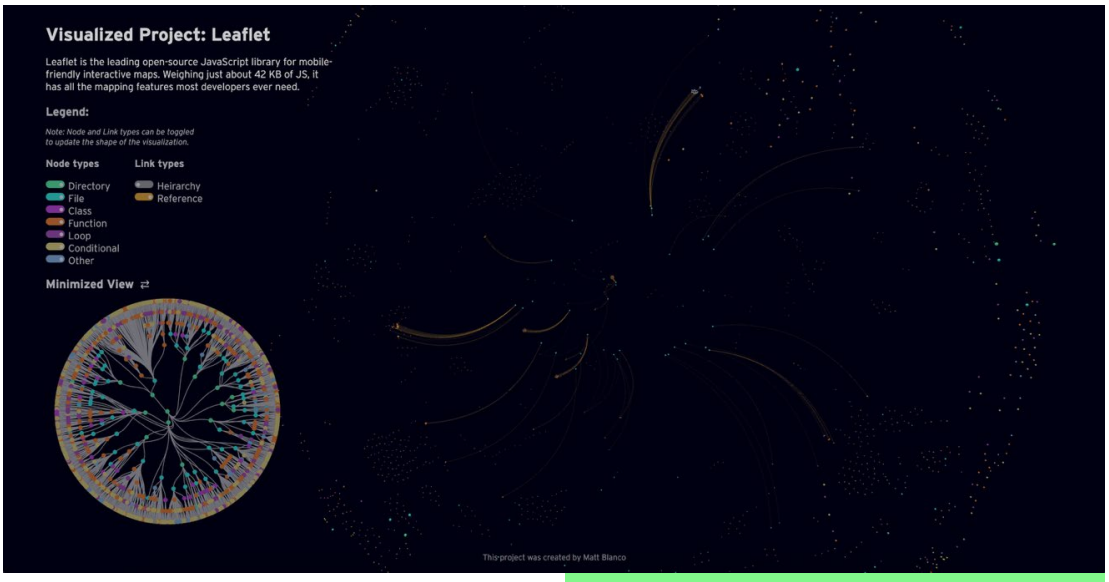

Fig. 4. Filtering out hierarchical edges through the toggle-able legend.

tion. Highlighting a node within the dendrogram will automatically move the user's camera to the node in the 3D network to edge the two visualizations. The user has the ability to change the views of the 3D and 2D visualizations if they prefer to interact with one over the other, and hovering over a node will highlight it in both visualizations to visually edge them. The added interactions and purposeful design attempt to improve readability when using the tool.The central focus of the tool is the 3D network shown front-and-center when first opening *Exploring Hidden Worlds*. Situated in the center of the visualization sits the root directory of the project, and expanding outwards are child directories, files, and eventually individual code statements. What would be a simple hierarchical tree diagram becomes more complex when referential edges are added to the network, turning the visualization into a more circular form.

The force-directed layout creates an engaging and dynamic animation as the visualization first loads and slowly cools down, inviting users to pan and zoom. Meanwhile, the visualization provides methods of information granularity by highlighting the first line of code or file name when hovering over nodes and displaying a tooltip of code that the node is representing.

Exploring the rendered codebase relies on a user's curiosity to navigate the visual representation of code at their own pace as opposed to providing an entry point to read the visualized data. Each part of the interface helps make navigating the visualization as easy as possible, and exploring the visualization creates a space for critical engagement with the infrastructure we are increasingly relying on. The goal is to use our technology thoughtfully and intently, or as Satyanarayan and Jones suggest:

Fig. 5. Information granularity when interacting with nodes.

> Within [a user's] capacity to act, they can strategically compose their own actions according to means–ends calculations and interpret the efficacy of their action on the world. Thus, each independent agent is engaged in an ongoing dialectical process of compositional and interpretive activity: acting with purpose and monitoring their action. (Satyanarayan and Jones 2024)

Through the visualization, a user can view the different design patterns present in a codebase, see clusters of nodes indicating which files are more complex, and see which files rely on other parts of a codebase. When paired with the information granularity within the visualization, a user can further investigate how specific portions of code work (Figure 5). The visualization provides a new material when it comes to understanding code, giving visual form to function. While reading individual lines of code with the 3D network is difficult, a user can better conceptualize the scale that software projects can take.

4. Conclusion

Software will continue to streamline all aspects of our day-to-day life, and augment the ways we think. Though it is our responsibility, as users of technology, to rethink how it gets utilized. There is a connection between having an understanding of digital technology through computational thinking and agency over software. However, passively relying on digital technology diminishes our agency, but it can be regained through critical engagement with code. The ongoing discourse on computational thinking and digital literacy underscores the necessity for novel representations to comprehend and interpret code among non-technical audiences. Information visualization is

particularly suited to shed light on the intricate nature of code due to its ability to translate information across domains of expertise and simplify complex information. Visualizations are also enhanced when combined with interaction design techniques by providing methods to explore visualized data such as data filtering. *Exploring Hidden Worlds* offers one solution to see how software is constructed by presenting a novel approach to visualize code. Each aspect of the interface, from the 3D network, filterable legend, and hierarchical 2D dendrogram, create a variety of interactions for a user to navigate the visualization, and the combination of design and programming methods used created an engaging interface for users to freely explore code. The project sought to develop an engaging method for learning and visually representing the digital material of code. As people gain a deeper understanding of how code influences their daily lives, through the interfaces we interact with and the data software collects, they can exert greater control over how digital technology seamlessly integrates into their lives.

Acknowledgements. We would like to thank Milena Kozlowska for her help and feedback while writing the paper. In addition to the Northeastern University Office of Undergraduate Research and Fellowship whose initial financial support led to this research endeavor.

References

Abbate, Lorenza, and Cristina Marino. 2022. "Designing Data Interaction in Exhibitions Contexts." *In DRS2022: Bilbao, 25 June - 3 July, Bilbao, Spain* edited by Dan Lockton, Sara Lenzi, Paul Hekkert, Arlene Oak, Juan Sádaba, Peter Lloyd. https://doi.org/10.21606/drs.2022.693.

Balzer, Michael, Oliver Deussen, and Claus Lewerentz. 2005. "Voronoi Treemaps for the Visualization of Software Metrics." *In Proceedings of the 2005 ACM Symposium on Software Visualization*, 165–72. ACM. https://doi.org/10.1145/1056018.1056041.

Chun, Wendy Hui Kyong. 2016. *Updating to Remain the Same: Habitual New Media.* The MIT Press. https://doi.org/10.7551/mitpress/10483.001.0001.

Cleveland, Harlan. 1982. "Information as a Resource." *The Futurist* 16 (6): 34–39.

Dimara, Evanthia, and Charles Perin. 2020. "What Is Interaction for Data Visualization?" *IEEE Transactions on Visualization and Computer Graphics* 26 (1): 119–29. https://doi.org/10.1109/TVCG.2019.2934283.

Filbert, Nathan W. 2021. "Learning the Code: Deciphering Digital Literacy." *International Journal of Digital Literacy and Digital Competence* 12 (2): 1–21. https://doi.org/10.4018/IJDLDC.287623.

George-Reyes, Carlos Enrique, Francisco Javier Rocha Estrada, and Leonardo David Glasserman-Morales. 2021. "Interweaving Digital Literacy with Computational Thinking." *In Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21)*, 13–17. ACM. https://doi.org/10.1145/3486011.3486412.

Guzdial, Mark. 2008. "Education Paving the Way for Computational Thinking." *Communications of the ACM* 51 (8): 25–27. https://doi.org/10.1145/1378704.1378713.

Hawes, Nathan, Stuart Marshall, and Craig Anslow. 2015. "CodeSurveyor: Mapping Large-Scale Software to Aid in Code Comprehension." In *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, 96–105. IEEE. https://doi.org/10.1109/VISSOFT.2015.7332419.

Heer, Jeffrey, and Ben Shneiderman. 2012. "Interactive Dynamics for Visual Analysis." *ACM Queue* 10 (2): 30–55. https://doi.org/10.1145/2133416.2146416.

Heinonen, Ava, Bettina Lehtelä, Arto Hellas, and Fabian Fagerholm. 2022. "Synthesizing Research on Programmers' Mental Models of Programs, Tasks and Concepts -- a Systematic Literature Review." arXiv. https://doi.org/10.48550/ARXIV.2212.07763.

Jbara, Ahmad, and Dror G. Feitelson. 2014. "JCSD: Visual Support for Understanding Code Control Structure". In *Proceedings of the 22nd International Conference on Program Comprehension*, 300–303. ACM. https://doi.org/10.1145/2597008.2597801.

Markus, Thomas A. 1993. *Buildings and Power*. Routledge. https://doi.org/10.4324/9781315003153.

Masud, L., F. Valsecchi, P. Ciuccarelli, D. Ricci, and G. Caviglia. 2010. "From Data to Knowledge - Visualizations as Transformation Processes within the Data-Information-Knowledge Continuum." In *2010 14th International Conference Information Visualisation*, 445–49. IEEE. https://doi.org/10.1109/IV.2010.68.

Satyanarayan, Arvind, and Graham M. Jones. 2024. "Intelligence as Agency: Evaluating the Capacity of Generative AI to Empower or Constrain Human Action." *An MIT Exploration of Generative AI*, March 27, 2024. https://doi.org/10.21428/e4baedd9.2d7598a2.

Stambler, Danielle Mollie, Saveena (Chakrika) Veeramoothoo, and Katlynne Davis. 2024. "Toward Digital Life: Embracing, Complicating, and Reconceptualizing Digital Literacy in Communication Design." *Communication Design Quarterly* 12 (2): 5–10. https://doi.org/10.1145/3655727.3655728.

Tamatea, Laurence. 2019. "Compulsory Coding in Education: Liberal-Humanism, Baudrillard and the 'Problem' of Abstraction." *Research and Practice in Technology Enhanced Learning* 14 (1): 14. https://doi.org/10.1186/s41039-019-0106-3.

Wettel, Richard, and Michele Lanza. 2007. "Visualizing Software Systems as Cities." In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 92–99. IEEE. https://doi.org/10.1109/VISSOF.2007.4290706.