From Handoffs to Co-Creation: Deepening Collaboration between Designers, Developers, and Data Science Workers in UX Design

K. J. KEVIN FENG, University of Washington, USA

AMY X. ZHANG, University of Washington, USA

UX design tools have made significant advancements in supporting real-time collaboration within design files that drastically shift the way teams discuss and iterate on designs. However, designs often still have little connection to the underlying technologies that power user interfaces when implemented, which limits how deeply designers can collaborate with technical non-designers. We propose new ways design tools can move from one-time handoffs towards supporting deeper *co-creation* between designers and technical stakeholders, specifically developers and data science workers, to create interactive and expressive interface designs. We do so through two main focus areas that connect code and data with design—bridging designer-developer divergence through component-based synchronization, and empowering designerly interaction with data to help prototype probabilistic interfaces. We conclude with a discussion on the implications of these two focus areas on future design tools.

CCS Concepts: • **Human-centered computing → Collaborative and social computing systems and tools**; **Systems and tools for interaction design**.

Additional Key Words and Phrases: design tools, collaboration, user experience

ACM Reference Format:
K. J. Kevin Feng and Amy X. Zhang. . From Handoffs to Co-Creation: Deepening Collaboration between Designers, Developers, and Data Science Workers in UX Design. In *InContext Workshop @ CHI 2022*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

In recent years, collaboration has been brought to the forefront of design tools [1, 12, 29]. These tools have taken great strides towards democratizing design for broader audiences: non-designers can now enter design files to look at designers' processes, comment on mockups, and even try their own hand at creating the designs they need. This also means that designers are more frequently collaborating with non-designers.

However, collaboration between designers and technical stakeholders is still oftentimes envisioned as a *handoff*, where a design file is iterated on by multiple stakeholders but then passed on to technical team members who take over implementation. Although design tools have released numerous features aimed at reducing friction during handoffs [5, 20, 28], collaboration between designers and developers can still be quite brittle, particularly as code implementations diverge from graphical mockups. At the same time, advancements in and excitement around data-intensive technologies such as machine learning are prompting them to be adopted in more user-facing applications. Design tools and processes are struggling to keep up in this area, offering little support for designers to envision and reason about data-driven interfaces.

We argue that a reconsideration of the collaboration away from a handoff towards *co-creation* of linked artifacts provides an opportunity for tool design to prompt deeper and more iterative collaboration between non-technical designers and technical stakeholders. By connecting designs and code via component-based synchronization, we can anticipate and bridge designer-developer divergence. Further, by connecting designs and data, we can enable designerly interactions with data and therefore enrich discussions with data science collaborators. Finally, we show our proposed concepts can exist harmoniously in a unified design environment.

2 TOWARDS DEEPER COLLABORATIONS WITH DEVELOPERS

The term "handoff" is frequently used to describe the process by which designers prepare their graphical work for developers to implement through code. Handoff is especially important in many software design workflows, where designers and developers often work independently [22]. Commonly used design tools (e.g. Figma, Sketch) have features that attempt to foster a smooth handoff, such as precise graphical specification of components through CSS and interactive, animated prototypes [5, 28]. With these features in place, however, collaborative breakdowns and duplicated work during handoff can still easily occur [19, 22, 38], resulting in diverging designs and implementation. In fact, this divergence can occur even with a perfectly coordinated handoff. As human-computer interaction researchers, we can have the luxury of working with a seamless handoff where we are both the designers and developers of a system, and even then, we may notice significant differences emerge between our designs and actual system as we progress along in our implementation.

Perhaps some issues surrounding handoff can be attributed to assumptions baked into the handoff process itself, which are twofold. First, the developer should aim to recreate in code what the designer has prepared, often in fine, even pixel-perfect[1], detail [20]. Second, as the name implies, handoff is often a *transfer of work* rather than a *co-creation* effort [20, 38]. This is reflected in current design tools: designs are to be "finalized", and in many cases "exported", in order for developers to be able to begin their implementation [5, 28]. Past research has attempted to address frictions through tools that blend code with visual design [19], as well tools that eliminate the need for handoff altogether by enabling designers to generate live user interfaces from their designs [8, 16, 18]. The latter is a vision that has still not yet come into fruition, despite being proposed by Landay and Myers in SILK more than 25 years ago [18].

Furthermore, it is important to note that previous work and attention around handoff has focused primarily on the stages of design and development leading up to the initial launch of the implementation. Iteration is a key component of the design process, but can be challenging to operationalize [40], and is made even more difficult if there is already divergence between design and implementation to begin with. There have been attempts to generate interface designs by vectorizing screenshots of deployed websites [37]. However, this was intended to help the designer during ideation and does not allow the designer to easily leverage their existing work. Instead of focusing on tools that aim to fully eliminate collaboration errors at handoff, we propose an alternate approach: keeping design and implementation mutually updated to minimize misalignment. It is important, then, to ask: **what can design tools look like to support robust but flexible linkages of design and code components *in expectation of design-development divergence*?**

We find encouragement in interactive frontend programming tools such as Storybook [31] that are also designer-friendly. Storybook allows users to create UI components through programmatic specification and then test and modify the component through direct manipulation. Indeed, designers have already started to experiment with Storybook as a tool for managing design systems [24]. However, this remains a developer-centered workflow as component authoring

[1]Designers use "pixel perfect design" to refer to a high standard of design that contains little to no graphical imprecisions [15].

is done via code by the developer. We envision an integrated workshop environment in which designers and developers can 1) design and prototype components through direct manipulation, 2) actualize components' interactive behaviors using code, and 3) collaboratively link the two and test for performance and edge cases. Note that this is a *component-first model* distinct from previous *interaction-first approaches* such as Enact [19]. We see value in a component-first model due to the long-standing emphasis on unit testing in software engineering [27], a shift towards component-based frontend programming frameworks [23, 36, 45], and the rise of component-based design systems in UX [9]. When designers assemble components into larger components and full UIs in the workshop environment, developers can see how the components will programmatically communicate with each other, paving the way for implementation. Changes to the component, whether it's the design or the code, can be indicated in the environment and reviewed by others to generate discussion around the artifact and offer decision transparency.

There are pitfalls, however. Designers and developers may find that this environment is too open and transparent, or the messiness of one type of work (ideation in design or bugs in programming) can negatively affect the other. We think this is an area worthy of further research.

3 TOWARDS DESIGNERLY INTERACTION WITH DATA

Advances and interest in artificial intelligence (AI), particularly AI methods based on machine learning (ML), means that data-driven technologies are increasingly being incorporated into user-facing applications. This poses some major challenges for UX practitioners. First, there is more potential for unpredictable and unexplained system errors [17, 41], leading to eroded user trust or even abandonment of the system entirely [10]. Second, ML's properties make it difficult to grasp as a design material—for example, a model's capabilities can vary depending on the data it was trained on and it may be difficult to envision ML implementations for a given UX problem [43]. Third, UX and ML workflows do not neatly align, and practitioners in both groups may experience substantial friction when collaborating [35]. Practitioners in both academia and industry have offered ways to ameliorate ML-specific design challenges. Subramonyam et al. introduced model-informed prototyping through ProtoAI [34], a tool that allows designers to integrate output from ML models into their prototyping work, and also suggested a process model for designers to better collaborate with AI engineers [35]. Companies have published human-AI interaction guidelines defining high-level best practices for user-facing AI products [2, 3, 13]. Although many challenges in this area still remain unaddressed, it is becoming clear that designers should be open to new tools and design methodologies in order to effectively design with ML.

Much of previous ML research has been grounded in model development and optimization [25]. As model architectures mature, there has been a recent shift in attention towards data-centric AI [33]: features of the training data—not the model itself—are systematically engineered for the ML system to achieve optimal performance [32]. Data-centric AI may in fact help streamline ML design by reducing the need to understand inner workings of (often complex) models, but still poses significant challenges due to the lack of data handling support in current design tools. More concretely, the *probabilistic and adaptive nature* of ML-enabled interfaces makes them especially difficult to design. Indeed, Yang et al. claims that ML systems presenting the highest levels of design difficulty are ones that constantly evolve and learn from new user data after deployment [43]. Being able to support probabilistic and data-adapting interface behaviors has benefits beyond ML, as ML isn't the only means to achieve such behaviors.

The data visualization community has developed a host of tools for interactive data analysis [7, 21, 26, 39, 47], many of which require no programming skills and may be integrated into design tools as third-party extensions. However, previous work examining how designers can design data visualizations point to a need for alternative approaches to data analysis in a design context [6, 21]. We hypothesize that designers will benefit tools that support *designerly* *interactions with data.* That is, understanding data in a way that aligns with existing mental models in the design process will help designers reason about adaptive UIs [46] whose form may change with end user data. For example, since user personas are a fundamental part of the design process [4], being able to cluster messy user data into distinct interaction profiles may aid designers in finding new design opportunities in their UIs. Bringing data into the design process may also help designers establish boundary objects [30] to collaborate with data science workers. A limited number of related works explored handoff-style design patterns designers can use to share their ideas with data scientists [44] but also found that designers are most effective in ML projects when they engage in sustained co-creation with data scientists [42]. However, it is unclear how designers can initiate such co-creation practices. Using tools that empower designerly interaction with data may be one way of doing so.

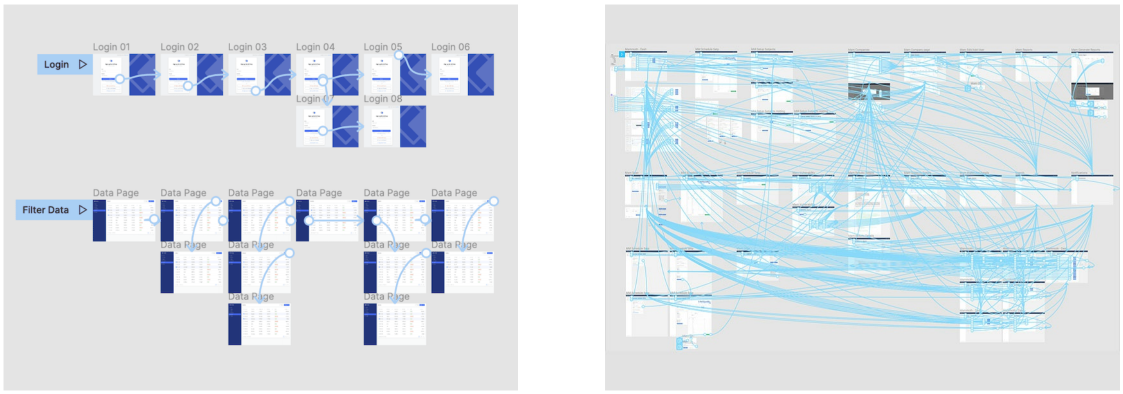
One of the fundamental differences between designing for deterministic systems and designing for stochastic ones is the need to shift from binary to probabilistic reasoning. For example, let us consider a designer who wishes to prototype the deterministic behavior of a close button. The designer creates a screen containing a modal pop-up menu with close button and links it to a screen without the menu via the click or tap of the close button. Now, we consider a designer who wishes to prototype the stochastic behavior of a dice rolling UI. The designer creates a screen containing a button that "rolls" a digital dice along with 6 outcome screens, one for each dice face. When the button is clicked or tapped, there will be a 1 in 6 chance of the user flow proceeding to a particular outcome screen. This example motivates the need for *probabilistic user flows* for stochastic UIs. Of course, not all data-intensive UIs are stochastic in nature (e.g. visualization interfaces). However, probabilistic user flows can be applied to a wide range of adaptive UIs—such as recommender systems and news feeds—to help designers simulate more realistic behaviors for user testing as well as envision more edge cases to design for.

4 IMPLICATIONS FOR FUTURE DESIGN TOOLS

We now illustrate, through an example, the functionality of a collaborative, component-based environment that supports designerly interactions with data. A designer mocks up a UI component that displays the results of a rider-driver match for a ride-hailing app. The developer adds frontend code to the component that implements its specified behavior. The designer then imports some ride match data provided by a data scientist into the tool, which lets the designer know that location is a potential factor that can trigger distinct user behaviors. The designer then feeds into the live component data from ride matches on a crowded city block, followed by matches from a suburban neighborhood, and finally matches from a rural town, to visualize component behavior for different user groups and test for edge cases. The designer realizes the empty state (in which there are no available drivers for the rider) is not handled well, and modifies the design accordingly. The change is signalled to the developer, who then updates the code for the empty state. The designer asks the data scientist for more information on the likelihood of empty state in various locations, with the intention of creating a probabilistic user flow to test the prototype with users in rural areas. The designer sets up the flow while the data scientist runs a statistical model to retrieve insights from existing data. Finally, the data scientist and designer update the user flow using findings from the analysis. The designer can now more accurately portray the app's matching behavior in rural areas and consequently acquire higher quality feedback from user testing with relevant populations.

Component-first collaboration and data-driven design may require unique considerations in future design tools. For example, it is common in the design process to start with rougher sketches or low-fidelity wireframes before refining them into higher fidelity prototypes [4]. In a collaborative workshop environment, one issue that may arise is developers prematurely implementing a component. If designers still need to make significant changes to a live component, developer collaborator(s) may need to rewrite significant portions of code or even toss out their previous efforts entirely. Designers should then have the ability to indicate to developers that no major behavioral changes will be made to a component. Additionally, if minor changes are made from either design or development later on, a review system should be in place for others to discuss, comment on, and accept or reject changes.

Probabilistic user flows also come with a caveat: they may result in complex, non-linear user flows. Consequently, designers' workspaces may quickly become cluttered to the point of being unmanageable, as prototyping features in current design tools are best suited for handling linear user flows (see Figure 1). Tools may need to consider new ways to organize information in non-linear user flows, such as using "screen families" to associate and collapse sets of related screens, such as the dice outcome screens in the earlier example.



(a) (b)

Fig. 1. (a) A simple linear user flow in Figma featuring a login and data filter screens (credit: Molly Hellmuth, "Ultimate Guide to Prototyping with Figma" on UIPrep [14]), and (b) a non-linear user flow in Figma for an app with more components and only slightly more screens (credit: Luke Dowding, "Prototyping with Figma" on Dribbble [11]).

5 CONCLUSION

UX design has become more collaborative, but collaboration can be so much more than inviting other stakeholders into a multiplayer design file. We argue that design tools can nurture deeper collaboration between technical collaborators, specifically developers and data science workers. With developers, synchronization of design and code components may anticipate design-development divergence and provide a playground for interactively testing, discussion, and review. We showed how this synchronization can open possibilities for designers to create probabilistic and adaptive UIs through designerly interactions with data. Yang et al. claim that existing design tools and prototyping methods are sufficient for simple probabilistic systems with limited outputs [43]. We showed that designers can benefit from additional tool support such as probabilistic user flows as well as closer collaboration with data stakeholders, such as data scientists.

We think this is a compelling avenue to explore the future of tools for richer collaboration between designers and non-design stakeholders more generally. We invite other workshop participants to engage in discussion on this model of component-based, data-driven collaboration, as well as related topics.

REFERENCES

[1] Adobe. 2022. Fast & Powerful UI/UX Design & Collaboration Tool. https://www.adobe.com/products/xd.html. Accessed: 2021-09-06.

[2] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for Human-AI Interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300233

[3] Apple, Inc. 2022. Introduction - Machine Learning - Human Interface Guidelines. https://developer.apple.com/design/human-interface-guidelines/machine-learning/overview/introduction/. Accessed: 2022-02-18.

[4] Jonathan Ball. 2005. The Double Diamond: A universally accepted depiction of the design process. https://www.designcouncil.org.uk/news-opinion/double-diamond-universally-accepted-depiction-design-process. Accessed: 2021-12-13.

[5] Joey Banks. 2022. Keeping files organized for your team. https://www.figma.com/best-practices/guide-to-developer-handoff/file-organization/. Accessed: 2022-02-16.

[6] Alex Bigelow, Steven Drucker, Danyel Fisher, and Miriah Meyer. 2014. Reflections on How Designers Design with Data. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces* (Como, Italy) *(AVI '14)*. Association for Computing Machinery, New York, NY, USA, 17–24. https://doi.org/10.1145/2598153.2598175

[7] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics* 17, 12 (2011), 2301–2309.

[8] Chunyang Chen, Ting Su, Guozhu Meng, Zhenchang Xing, and Yang Liu. 2018. From UI Design Image to GUI Skeleton: A Neural Machine Translator to Bootstrap Mobile GUI Implementation. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 665–676. https://doi.org/10.1145/3180155.3180240

[9] Elizabeth F Churchill. 2019. Scaling UX with design systems. *Interactions* 26, 5 (2019), 22–23.

[10] Maartje de Graaf, Somaya Ben Allouch, and Jan van Dijk. 2017. Why Do They Refuse to Use My Robot? Reasons for Non-Use Derived from a Long-Term Home Study. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction* (Vienna, Austria) *(HRI '17)*. Association for Computing Machinery, New York, NY, USA, 224–233. https://doi.org/10.1145/2909824.3020236

[11] Luke Dowding. 2019. Ultimate Guide to Prototyping in Figma. https://dribbble.com/shots/5959155-Prototyping-with-Figma. Accessed: 2022-02-22.

[12] Figma. 2022. Creative tools meet the internet. https://www.figma.com/about/. Accessed: 2022-02-15.

[13] Google PAIR. 2022. People + AI Guidebook. https://pair.withgoogle.com/guidebook/. Accessed: 2022-02-18.

[14] Molly Hellmuth. 2021. Ultimate Guide to Prototyping in Figma. https://www.uiprep.com/blog/ultimate-guide-to-prototyping-in-figma. Accessed: 2022-02-22.

[15] Bohyun Kim. 2013. Responsive web design, discoverability, and mobile challenge. *Library technology reports* 49, 6 (2013), 29–39.

[16] Ju-Whan Kim and Tek-Jin Nam. 2013. EventHurdle: Supporting Designers' Exploratory Interaction Prototyping with Gesture-Based Sensors. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 267–276. https://doi.org/10.1145/2470654.2470691

[17] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces* (Atlanta, Georgia, USA) *(IUI '15)*. Association for Computing Machinery, New York, NY, USA, 126–137. https://doi.org/10.1145/2678025.2701399

[18] James A. Landay and Brad A. Myers. 1995. Interactive Sketching for the Early Stages of User Interface Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '95)*. ACM Press/Addison-Wesley Publishing Co., USA, 43–50. https://doi.org/10.1145/223904.223910

[19] Germán Leiva, Nolwenn Maudet, Wendy Mackay, and Michel Beaudouin-Lafon. 2019. Enact: Reducing Designer–Developer Breakdowns When Prototyping Custom Interactions. *ACM Trans. Comput.-Hum. Interact.* 26, 3, Article 19 (may 2019), 48 pages. https://doi.org/10.1145/3310276

[20] Oliver Lindberg. 2019. Best Practices for a Smoother Designer-Developer Handoff. https://xd.adobe.com/ideas/perspectives/leadership-insights/guide-to-smooth-designer-developer-handoffs/. Accessed: 2022-02-19.

[21] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. *Data Illustrator: Augmenting Vector Design Tools with Lazy Data Binding for Expressive Visualization Authoring.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173697

[22] Nolwenn Maudet, Germán Leiva, Michel Beaudouin-Lafon, and Wendy Mackay. 2017. Design Breakdowns: Designer-Developer Gaps in Representing and Interpreting Interactive Systems. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing* (Portland, Oregon, USA) *(CSCW '17)*. Association for Computing Machinery, New York, NY, USA, 630–641. https://doi.org/10.1145/2998181.2998190

[23] Meta Platforms, Inc. 2022. React—A JavaScript Library for Building User Interfaces. https://reactjs.org/. Accessed: 2022-02-19.

[24] Taylor Palmer and Jordan Bowman. 2021. 2021 Design Tools Survey. https://uxtools.co/survey-2021/. Accessed: 2022-01-25.

[25] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "Everyone Wants to Do the Model Work, Not the Data Work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 39, 15 pages. https://doi.org/10.1145/3411764.3445518

[26] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 341–350.

[27] Koushik Sen, Darko Marinov, and Gul Agha. 2005. CUTE: A Concolic Unit Testing Engine for C. In *Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Lisbon, Portugal) *(ESEC/FSE-13)*. Association for Computing Machinery, New York, NY, USA, 263–272. https://doi.org/10.1145/1081706.1081750

[28] Sketch B.V. 2022. Developer Handoff. https://www.sketch.com/docs/developer-handoff/. Accessed: 2022-02-19.

[29] Sketch B.V. 2022. Work together. Go far. https://www.sketch.com/collaborate/. Accessed: 2021-12-14.

[30] Susan Leigh Star and James R Griesemer. 1989. Institutional ecology,translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social studies of science* 19, 3 (1989), 387–420.

[31] Storybook. 2022. Build component driven UIs faster. https://storybook.js.org/. Accessed: 2022-02-16.

[32] Eliza Strickland. 2022. Andrew Ng: Unbiggen AI. https://spectrum.ieee.org/andrew-ng-data-centric-ai. Accessed: 2022-02-18.

[33] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. How Can Human-Centered Design Shape Data-Centric AI? (2021), 3 pages.

[34] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. *ProtoAI: Model-Informed Prototyping for AI-Powered Interfaces*. Association for Computing Machinery, New York, NY, USA, 48–58. https://doi.org/10.1145/3397481.3450640

[35] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. 2021. *Towards A Process Model for Co-Creating AI Experiences*. Association for Computing Machinery, New York, NY, USA, 1529–1543. https://doi.org/10.1145/3461778.3462012

[36] Svelte. 2022. Svelte · Cybernetically Enhanced Web Apps. https://svelte.dev/. Accessed: 2022-02-19.

[37] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Amy J. Ko. 2018. *Rewire: Interface Design Assistance from Examples*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174078

[38] Jagoda Walny, Christian Frisson, Mieka West, Doris Kosminsky, Søren Knudsen, Sheelagh Carpendale, and Wesley Willett. 2020. Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 12–22. https://doi.org/10.1109/TVCG.2019.2934538

[39] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2015. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 649–658.

[40] David C Wynn and Claudia M Eckert. 2017. Perspectives on iteration in design and development. *Research in Engineering Design* 28, 2 (2017), 153–184.

[41] Qian Yang, Justin Cranshaw, Saleema Amershi, Shamsi T. Iqbal, and Jaime Teevan. 2019. Sketching NLP: A Case Study of Exploring the Right Things To Design with Language Intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300415

[42] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. 2018. Investigating How Experienced UX Designers Effectively Work with Machine Learning. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 585–596. https://doi.org/10.1145/3196709.3196730

[43] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. *Re-Examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376301

[44] Qian Yang, John Zimmerman, Aaron Steinfeld, and Anthony Tomasic. 2016. Planning Adaptive Mobile Experiences When Wireframing. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 565–576. https://doi.org/10.1145/2901790.2901858

[45] Evan You. 2022. Vue.js - The Progressive JavaScript Framework. https://vuejs.org/. Accessed: 2022-02-19.

[46] John Zimmerman, Changhoon Oh, Nur Yildirim, Alex Kass, Teresa Tung, and Jodi Forlizzi. 2020. UX designers pushing AI in the enterprise: a case for adaptive UIs. *Interactions* 28, 1 (2020), 72–77.

[47] Jonathan Zong, Dhiraj Barnwal, Rupayan Neogy, and Arvind Satyanarayan. 2020. Lyra 2: Designing interactive visualizations by demonstration. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 304–314.