Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff

Jagoda Walny, Christian Frisson, Mieka West, Doris Kosminsky,
Søren Knudsen, Sheelagh Carpendale, Wesley Willett

Abstract— Complex data visualization design projects often entail collaboration between people with different visualization-related skills. For example, many teams include both designers who create new visualization designs and developers who implement the resulting visualization software. We identify gaps between data characterization tools, visualization design tools, and development platforms that pose challenges for designer-developer teams working to create new data visualizations. While it is common for commercial interaction design tools to support collaboration between designers and developers, creating data visualizations poses several unique challenges that are not supported by current tools. In particular, visualization designers must characterize and build an understanding of the underlying data, then specify layouts, data encodings, and other data-driven parameters that will be robust across many different data values. In larger teams, designers must also clearly communicate these mappings and their dependencies to developers, clients, and other collaborators. We report observations and reflections from five large multidisciplinary visualization design projects and highlight six data-specific visualization challenges for design specification and handoff. These challenges include *adapting to changing data*, *anticipating edge cases in data*, *understanding technical challenges*, *articulating data-dependent interactions*, *communicating data mappings*, and *preserving the integrity of data mappings across iterations*. Based on these observations, we identify opportunities for future tools for prototyping, testing, and communicating data-driven designs, which might contribute to more successful and collaborative data visualization design.

Index Terms—Information visualization, design handoff, data mapping, design process

1 Introduction

Creating visualizations is a challenging, multifaceted problem that requires a combination of skills and tools for data analysis, design, and development. Designers and developers must gain an understanding of the dataset and its characteristics through data exploration, then design data mappings, aesthetics, and interactions based on it [6]. These designs also need to be realized and deployed, typically by writing software. Sometimes it is possible for one person to perform all of these activities given enough time and resources. However, for more complex visualization projects with limited timelines, it is more feasible to distribute these activities amongst people in specialized roles.

This distribution of roles creates the challenge of *handoff*, the codifying and exchange of information between people working on different roles in a project, and the related challenge of communicating domain knowledge across roles. This problem is already well-known in general software design, where interaction designers are often distinct from software developers [39]. Over the past two decades, a wide range of specialized tools has emerged to help interaction designers outline and prototype interfaces in ways that reduce the friction between graphical designs and code. Commercial tools like Adobe XD [2], InVision [27], and Sketch [9] support expressive and precise visual design; interactive prototyping of animations, transitions, and interactions; exporting specifications during development and assets for developed applications.

- *All authors are with the University of Calgary, Calgary, Alberta, Canada.*
- *Christian Frisson's secondary affiliation is with McGill University, Montreal, Quebec, Canada. E-mail: christian@frisson.re.*
- *Doris Kosminsky's primary affiliation is with Federal University of Rio de Janeiro, Rio de Janeiro, Brazil. E-mail: doriskos@eba.ufrj.br.*
- *Søren Knudsen's secondary affiliation is with University of Copenhagen, Copenhagen, Denmark. E-mail: sknudsen@ucalgary.ca.*
- *Sheelagh Carpendale's primary affiliation is with Simon Fraser University, Vancouver, British Columbia, Canada. E-mail: sheelagh@sfu.ca.*
- *Remaining author e-mails: {jkwalny, mieka.west, wesley.willett}@ucalgary.ca.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Unfortunately for visualization designers, these tools lack robust support for data-driven designs. In practice, many programming-literate visualization designers still work largely in code, exploring datasets through iterative prototyping using libraries like D3 [10] or notebook environments like Observable [47]. We call this *design-as-development*. However, using these low-level tools requires considerable technical skills and can increase the time and effort needed to articulate, refine, and polish visualization designs. This scales poorly for large visualization projects, which may involve not just developers but also interaction designers, data experts, and clients, each with their own tool sets and institutional processes. In these collaborative settings, differing design

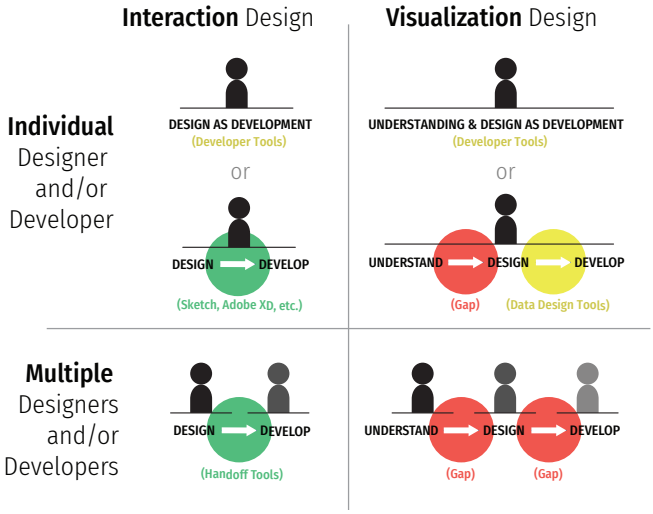

Fig. 1: Contemporary interaction design tools increasingly enable smooth transitions and collaboration between design and development (top-left) and handoffs between designers and developers (bottom-left). In our experiences across projects, these transitions remain challenging for visualization designers (top-right) and teams (bottom-right).

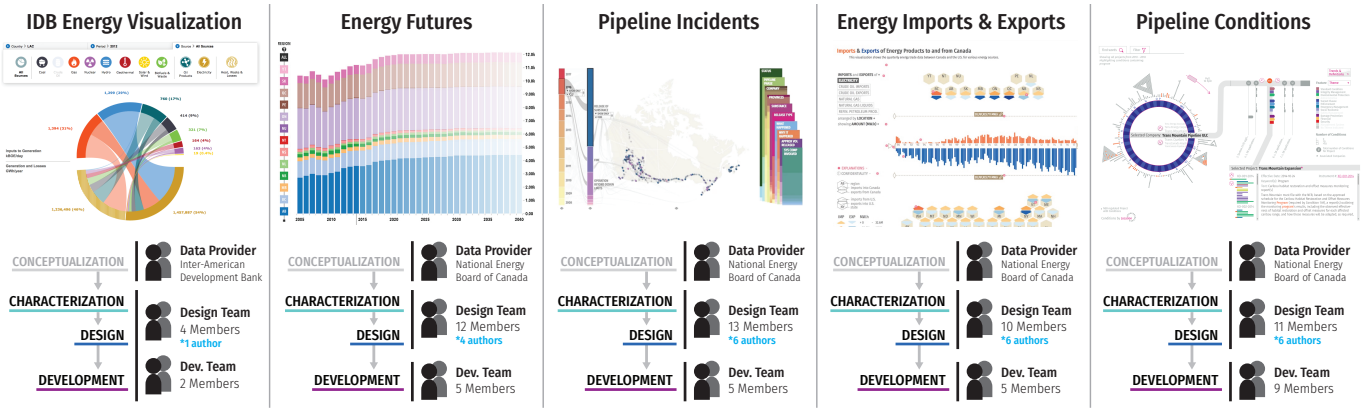

Fig. 2: The five visualization design and development projects in which we ground our observations.

objectives and gaps in the tools used to characterize data, design visualizations, and develop applications can exacerbate the issues caused by a lack of data-driven support. As with any sizeable software project, it is important to get things right during the early stages to avoid costly redesigns late in the project. While unexpected issues can be handled with some agility by small, flexible teams, these issues are easily amplified in larger teams. Larger teams tend to require input from more people, resulting in higher organizational overhead. This can put a strain on resources, shift timelines, and lead to sub-optimal results.

We draw on our experience as part of the design team on five complex data visualization projects (Figure 2) intended for wide-scale public release. Across each of these projects, we observed and participated in interactions between teams of designers and developers working together and separately to characterize data, create initial designs, and translate those designs into production-ready applications. We draw on those experiences to highlight challenges and opportunities specific to designer-developer collaboration in data visualization design projects.

While others before us have discussed practical visualization design projects [31, 60] and design studies [55], our focus is different. We focus on the practical work and coordination that goes into building visualizations. Although our projects did employ several researchers, they also employed practitioners: designers and developers. Our projects also involved close coordination with project coordinators and data experts on the data provider's side. This complex structure and the physical and temporal separation of different teams heightened the visibility of several practical challenges still faced during the visualization design process. Our work articulates issues that are of a very practical nature and that we expect are frequently experienced by others who are doing practical visualization work. We think our contributions add value to this practitioner-oriented research space, especially in light of the visualization research community's recent focus on practitioners as a crucial source of *"energy, ideas, and application problems"* [23].

2 Roles in Visualization Design Processes

Visualization design and development requires several unique sets of skills including experience in human-centered design, perception, evaluation, statistics, and graphics programming [31]. This conventional wisdom is exemplified by the (somewhat mythical) notion of "full-stack" visualization designer-developers capable of conducting the full range of "data wrangling, dynamic graphics, and derring-do" [22]. However, real-world visualization design projects (especially large ones) often include a variety of team members with diverse and overlapping subsets of these skills. As projects grow, these teams can become segmented, with responsibility for design and development delegated to individuals or teams whose skill sets and preferred tools can be increasingly disjoint. In particular, institutional and disciplinary divides can result in the partitioning of early-stage *design* tasks — such as data understanding, ideation, creating mockups, and prototyping — and *development* tasks like implementation, testing, deployment, and maintenance.

Disciplinary divides between designers and developers can be stark [39]. The interaction design literature has examined the divide between designers and developers from a number of angles, including: how designers and developers align their work in collaborations [11]; how they work remotely [65]; and how design tools can function as boundary objects that could mitigate designers' lack of "material" experience with software [49]. Recent work by Maudet et al. [39] has drawn attention to *design breakdowns* in design handoff, in which potential disconnects between designers and developers are highlighted by difficulties in implementing the final design. Leiva et al. [35] expand on this concept, identifying several specific types of breakdowns — including *omitting critical details*, *ignoring edge cases*, and *disregarding technical limitations* — that routinely contribute to difficulties in projects involving handoffs between designers and developers.

However, research in data visualization design still often fails to acknowledge this division of design and development. For example, McKenna et al.'s Design Activity Framework [41] combines design and implementation into a single "make" step and assumes that the responsibility for both will be tightly integrated. Other reflections on visualization design practice also tend to share this assumption, drawing primarily on the perspectives of visualization design researchers tasked with *both creating and implementing* novel visualizations as part of bigger multidisciplinary teams [31, 55, 60].

The design and development of new visualizations, like that of other interactive systems, entails considerable iteration and involves transitions between multiple sets of tools as designs move from conception to implementation. While some amount of visualization design and development often happens via coded prototypes, many aspects of visualization design — from early-stage concept generation and sketching through to late stage aesthetic refinement — are often better-served by graphic design and interaction design tools that offer greater expressive flexibility, as evidenced by the work practices of data visualization designers interviewed by Bigelow et al. [6]. However, unlike most other work in interaction design, the form of new visualizations depends intrinsically on the data that they will communicate. As a result, the process of visualization design is often a complex and iterative one anchored in multiple rounds of data examination, ideation, creation, and deployment [41]. These activities pose challenges for designers who may need to transition repeatedly between interactive tools that allow them to examine data and explore a diverse range of designs and more low-level data-driven development and coding. These issues are compounded as projects grow and responsibilities for design, development, and deployment are divided across multiple individuals or teams, each with different skill sets and priorities. In these situations, visualization design and development become an exercise in co-creation [6], complicated by dependencies between teams and differences in their competencies. Large diverse teams make it possible to create, deploy, and provide long-term support for complex visualizations. However, this division of labor reveals a variety of new design handoff and iteration challenges, which can be exacerbated by the data-driven nature of visualization design.

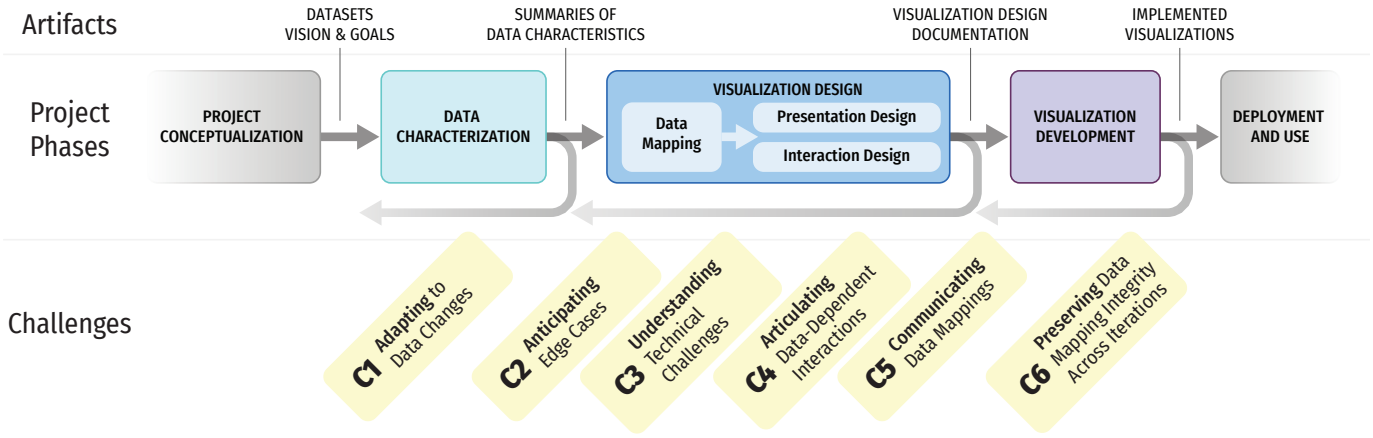

Fig. 3: Stages of a data visualization development process and the dependencies between them. Artifacts are produced that aid in the documentation and communication of the work done at each stage. Colored stages are reflected in this paper and relate to the challenges in yellow.

3 Overview of Visualization Design Projects

Our reflections on handoff in visualization design and development are anchored in our own experiences as members of a design team on five large data visualization design projects (Figure 2) conducted between 2012 and 2019. Each project involved between six months and several years of data characterization, design, and development work.

For each project, the work was directed by an outside client who was also the data provider. Our multi-member **design teams**, which included a rotating cast of designers, visualization researchers, postdocs, graduate students, and interns, were responsible for the majority of the data characterization and visualization design. In all projects, at least one (and typically more) of the authors participated in the process directly as members of the design team. A separate **development team** was tasked with creating, deploying, and providing initial maintenance for the final web-based applications.

3.1 Projects

Energy Visualization for the Inter-American Development Bank (IDB). The earliest of the projects, conducted between 2012–16 with the Inter-American Development Bank produced a suite of visualizations showcasing energy source generation, import and export, transmission, and consumption for countries in the Americas, as well as other benchmark countries. The resulting visualizations were hosted publicly from 2013–18, but are no longer accessible as of 2019. In this project, the design and development teams were more closely integrated than in the other projects, with both playing a substantial role in data characterization, design, and development.

Energy Futures. This project, conducted over 4 months in 2016, led to the development of four visualizations based on forecasts of Canadian energy production and consumption [8]. A second 7-month iteration [32] of the project in 2017 added a fifth visualization showcasing changes in projected energy demand across the Canadian provinces and territories. The visualizations are publicly available at https://apps2.neb-one.gc.ca/dvs.

Pipeline Incidents. Developed during 2017, this 8-month project produced an interactive visualization system that supported visual exploration of incidents that occurred on or around federally-regulated pipelines. The visualization is publicly available at https://apps2.neb-one.gc.ca/pipeline-incidents/.

Energy Imports & Exports. Another similarly-scoped project conducted over 16 months in 2017–18 involved creating a set of five visualizations showing historical imports and exports of various energy products from Canada. The visualizations are publicly available at https://apps2.neb-one.gc.ca/imports-exports/.

Pipeline Conditions. The most recent project, conducted over 18 months during 2018–19, focuses on visualizing the conditions placed by government regulators on the construction of new pipelines. At the time of publication, this project was near completion, but not yet publicly accessible.

3.2 Design Team Roles

The members of the design team needed to fulfill a variety of design-related roles. The project needed team members who could **characterize data**, including data wrangling, exploring data in existing visualization tools, spreadsheets, or code, processing data (including text mining), and understanding specific data types (for example, linguistic analysis of text data). All team members needed to **create and understand data mappings** from data to visual representation, which included varying degrees of ideation, basic perceptual understanding, and applying knowledge of visual variables. The project also required people with **visual design** skills who could design graphics, page layout, and typography while keeping accessibility in mind. The team also included people with skills in **interaction design**, including skills in prototyping and animation. Likewise, some team members **developed visualization prototypes** to verify and demonstrate designs and engineered the technically complex portions of design documents. All team members needed to **collaborate** and **communicate** with the data provider and development teams. All of these skills were complemented by knowledge of appropriate use of existing design, visualization, and development tools, as well as adapting to potential new tools.

During the Inter-American Development Bank project, the design team consisted of one primary visualization design researcher and three Visual Arts students (one undergraduate and two graduate students). The development team consisted of one primary computer science researcher and one doctoral student. These teams worked closely together in an iterative fashion and were located on the same campus.

In the remaining projects, the design and development teams were separate. The design team consisted of two primary investigators (visualization researchers), one project coordinator, one design researcher, 1-2 postdoctoral visualization researchers, 2-3 undergraduate or recently graduated computer science students, 0-1 information design undergraduate students, and 1-4 full-time employees with roles in design, development, and specialized data understanding. The development team consisted of anywhere between 5 and 9 members of a professional software development firm located in the same city. In all cases, the data providers were from separate institutions and were physically separated from the design and development teams.

3.3 Analysis and Synthesis Process

We identified the data-related challenges described in this paper via an ongoing process of reflection [44] through which we worked to refine our design team's work and communication practices. During each project, we kept records of artifacts produced for meetings, data explorations, ideation sketches, planning timelines, and design documents. We also regularly reflected on communication and design challenges

within our own team. Throughout, we documented and scrutinized the process using approaches drawn from diary-based [15, 48, 53, 56] and autobiographical studies [12, 17, 46]. Individual members of the design team, as part of their personal practice, kept notes and images documenting their work. Later, as part of this autobiographical process, we carefully re-examined our diary-based records and used them to identify gaps and challenges.

Starting with the Pipeline Incidents project, we also took steps to formalize our design communication processes. Within the design team, we leveraged our initial observations to create shared tools and processes for tracking the team's progress. During the implementation phase of each project, we also held face-to-face design review meetings with all stakeholders present. Finally, after each of the three last projects, we organized formal process discussions with the data provider and with members of both the design and development teams to help improve coordination for subsequent projects. We took detailed collaborative notes at all meetings.

Based on these reflections, we focused increasing energy across the remaining projects on improving design communication both within and across the teams. As part of this effort we documented meetings and design processes using detailed personal records, team records, design handoff documents, and handoff document revisions. We teased apart the details of the challenges presented in this paper by drawing on these detailed records. In discussing a particular challenge we could rigorously re-examine the time-stamped process by which each design was created, handed-off, implemented, re-discussed, re-implemented, and ultimately released.

Throughout our reflection, we noticed that a number of the recurring handoff challenges were not merely interaction design issues (like those documented by Leiva et al. [35]), but were instead rooted in the deep dependence of the designs on data. From these reflections, we have synthesized the most prominent unresolved data-related challenges and illustrated them using real examples from our projects. Where possible, the initial reflection was written by the team member who most closely experienced the example issue.

4 Visualization Design & Development

One outcome of our reflection on the processes and communication in our design projects is a formal model of the major phases of our design projects (Figure 3). This model was borne out of a need for a vocabulary to use when coordinating with multiple parties, and serves as a useful anchor for the design and communication challenges we discuss in the remainder of this paper. In a designer-as-developer scenario, a single person or small team might carry out all of these phases with less need for a formal process. In contrast, in our scenarios, which often involved multiple teams who did not share a daily working space, this model emphasizes the distribution of roles. In addition, it highlights the kinds of artifacts that can often facilitate communication across phases.

Project Conceptualization. This phase occurred on the client side, prior to the direct involvement of the design team. The client provided the design team with the ***vision and goals*** for the project as well as a ***dataset***. The handoff of these artifacts ranged from simple emailed delivery to more involved full-day workshops between client-side data experts and the design and development teams.

Data Characterization. In this phase, the design team explored and characterized the data, prioritizing analyses motivated by the project vision. This included understanding data types, amounts, and extrema; and the relation of each data facet to the project goals. In some cases, the design team recommended a more focused dataset for the visualization, which might include several iterations with the client-side data team. This phase sometimes involved data wrangling [29], but was more akin to exploratory data analysis or domain problem characterization [45]. We used a number of tools to support this phase, including hand-coded scripting, spreadsheets, visualization exploration software such as Tableau [58], and hand-sketching for preliminary ideation. This culminated in the creation of a ***summary of data characteristics***, which usually came in the form of an in-person presentation to the client-side data experts for verification. The presentation and the knowledge gained throughout this phase served as a foundation phase.

Visualization Design. The design phase encompassed the abstraction and encoding/interaction design phases of the nested model [45] together with partial algorithm design [45] and extensive visual presentation design. This was a two-stage process: first, we developed a concept for the visualization to be approved by the client. Then, we refined and polished the final design and documented it in the ***visualization design documentation*** shared with both clients and developers. We developed a *data mapping* on the basis of the data characterization relying heavily on hand-sketching and manual illustration in tools like Adobe Illustrator [1]. In some cases, we used chart generation tools and utilities (e.g. RAWGraphs [40] and Color Brewer [25]) and hand-coded prototypes using libraries such as D3 [10]. We developed the *presentation design* — the overall size and layout of the visualization — primarily using Adobe Illustrator [1]. Furthermore, we designed the *interaction* using various tools including paper prototyping [52], textual and sketched descriptions, and some general-purpose interaction prototyping tools (e.g. Axure [4] and Atomic [3]). The final development document was initially in PDF form but later evolved to be a web-based document. The most recent design document was based on Idyll [14], which allowed us to combine mockups, coded prototypes, and explanatory text in a single document.

Visualization Development. This phase was led by the software development team. As the design team, our role was mainly reactionary - we responded to questions about the design, suggested redesigns when issues arose, and verified that the implementation was functioning as intended. Most of the discussions around this phase were grounded in the design documentation as well as increasingly polished iterations of the ***implemented visualizations***.

Deployment and Use. As the visualization was deployed for public use, the software development team was tasked with its maintenance, including implementing quarterly data updates. The design team was involved if a data update contained unexpected values that were not supported by the existing design.

5 Challenges when Designing with Data

Based on our reflection and observation, we describe six gaps in the data visualization design process.

C1. Adapting to Data Changes

Data updates can have cascading effects on the data characterization, visualization design, and development phases because all aspects of visualization development depend upon the data. The impact of such effects may not be clear to data providers.

In our experience, data is rarely available in its full and final form before the visualization development process begins, often necessitating data updates later in the process — sometimes even post-deployment. Data changes are particularly impactful if they change the data characterization or the data used to generate views in the visualization design. Even when a data change is seemingly innocuous and does not change the general data mapping, it may affect the implementation stage, particularly where server-side mechanisms for loading, aggregating, or preparing data have already been established. For example changing a column name or unit symbol may break existing data parsers.

Data updates are not necessarily undesirable. They might provide corrections or additional data, or they might reflect a positive evolution of how the data provider releases data for the visualization. Such evolution might itself be prompted by witnessing the interim results of the visualization design process. As such, the challenge is not to avoid data updates altogether but to be able to cope with them efficiently.

For example, late in the process of designing visualizations for the *Inter-American Development Bank* project, the design and development teams had created a mature, late-stage visualization design of energy generation data from Latin American countries (Figure 4-top). Up to that point, all design decisions had been made on the basis of the team's work with the initial data provided by the client. This characterization led to a design that arranged data about different energy sources on a circle, showing the relationship between energy inputs (on the top half of the circle) to energy generation and losses (on the bottom half).

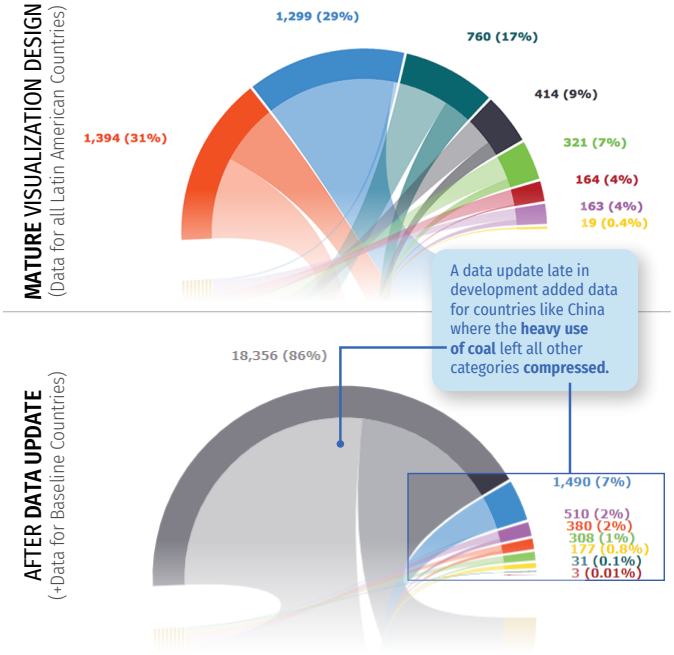

Fig. 4: Adding data from additional countries late in the design of this visualization resulted in a single energy source (coal) dominating the view which made other energy sources difficult to compare.

At this stage, an update added data for benchmark countries like China, creating views (Figure 4-bottom) in which a single energy source (in this case coal) visually overwhelmed the values from other sources. This dramatically altered the form and legibility of the visualization, crowding all of the original data onto a small slice at the side of the chart, and overlapping the arcs and labels. Given the late stage of the design process, there were not enough resources available to iterate the design, and the resulting visualization was quite different from the original concept.

Late-stage data updates can also cause subtle changes to how a visualization is perceived. For example, in the *Energy Futures* project, a visualization of demand shares by energy source was initially designed using data based on the energy production stages. The design team characterized the data and moved on to the design phase, creating a D3 prototype (Figure 5). Our focus here was to support comparison between provinces that have order-of-magnitude differences in scale. Later, the data was changed to a set based on end-use energy demand. The characteristics of both data sets were quite similar, so the design work continued with attention shifting to other concerns. However, there was a key but subtle difference in the new dataset — nearly all renewable energy was included within the Electricity category rather than in the Renewables category. The Renewables category in the original dataset was already quite small, so this change went unnoticed. While the resulting visualization still shows the data accurately, it was not designed to communicate the fact that the Electricity category include renewables as well.

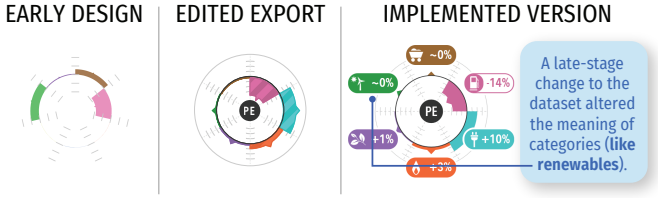

Fig. 5: Changes to the dataset for this *Energy Futures* visualization altered category meanings, creating potentially misleading values.

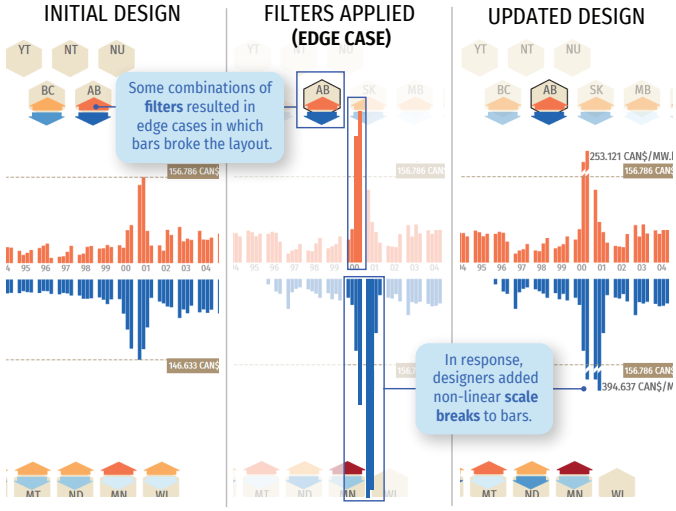

Fig. 6: The initial design of the *Energy Imports & Exports* visualization (left) responded poorly to particular combinations of filters (center) and ultimately required a revision (right).

C2. Anticipating Edge Cases

It is difficult for designers to anticipate and test all possible combinations of interactive inputs that a visualization might receive. As a result, it can be hard to anticipate situations in which a chart design or data mapping may break.

Common visualization interactions such as filtering or aggregation effectively change data mappings in real time. In design-as-development scenarios with live prototypes, these kinds of interactions can be tested relatively early in the development process and designs can be adjusted as needed. However, when design and development are separated, designers do not always have the tools or skillsets to fully test all possible combinations of inputs. As such, potential problems might only be uncovered during the development phase after many design decisions have been finalized. At this stage, any changes to the design can incur significant design and development work, limiting the possible ways in which the visualization design can be adapted to mitigate the problems.

In the design of the *Energy Imports & Exports* visualizations, the design team developed a mirrored chart that could be filtered to show average quarterly electricity prices between any combination of US states and Canadian Provinces (Figure 6-left). This data mapping worked well for the vast majority of views, including the various combinations of test data that the design team used when creating their initial documents. However, once this design was implemented it became clear that filtering by particular combinations of states and provinces revealed outliers which had been masked in the initial samples (Figure 6-center).

The design solution was constrained by the fixed size and very limited space available for the bar chart, as well as by the need to maintain consistency with the data mapping in other parts of the visualization. Reconfiguring other parts of the visualization was also not feasible late in the development stage. Ultimately, the design team opted to use a compressed scale break for these outliers (Figure 6-right). This solution makes it impossible to make direct visual comparisons between large values above the scale break and reduces the visual impact of large values, but still communicates relative differences in scale within the available space and minimized changes in other parts of the visualization. Several related scale issues also emerged late in the development of the *Pipeline Conditions* visualization, necessitating new design work during implementation. In all cases, if the edge cases had been identified earlier in the design process, the entire design might have been conceived differently.

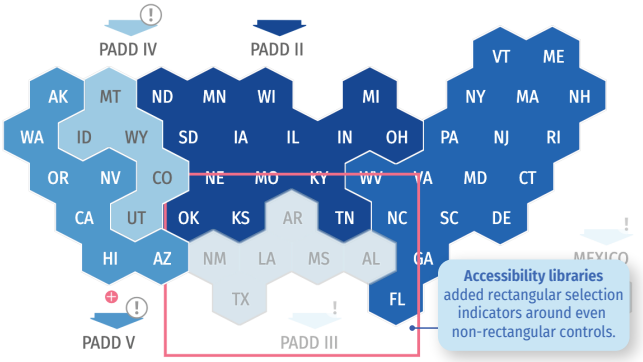

Fig. 7: A portion of a visualization showing a heat map of Canadian energy exports to United States Petroleum Administration Defense Districts (PADDs). Due to confusion surrounding the limitations of an accessibility template, color mappings and the layouts of non-rectangular components in this visualization needed to be adapted to accommodate rectangular selection indicators.

C3. Understanding Technical Constraints

Designers may not be aware of all of the technical constraints and challenges that can occur during the development phase. This leads to uncertainty about design feasibility and can trigger more dramatic revisions during development.

It is not always clear what types of software and hardware limitations may pose challenges for a design. Visualization designers tend to focus considerable attention on the choice of visual mappings and on providing a useful and appealing interaction experience. Web and application developers, meanwhile are more likely to be tasked with delivering robust, efficient, standards-compliant implementations of a design. While collaborators often have an appreciation for the others' areas of focus, it can be difficult to be aware of all potential issues that the other group faces. From a design perspective, this can lead to uncertainty, particularly when proposing unique or unconventional designs. Across our projects, the designers often tried to mitigate this uncertainty by prototyping and testing novel pieces of the designs in code. However, we still frequently encountered technical hiccups during development.

For example, during the design of the *Pipeline Conditions* visualization the design team created detailed working versions of several complex components, including an interactive keyword browser that leveraged a third-party physics engine. However, the development team worked within a different set of constraints that included cross-browser compatibility, future code maintainability, and a decision to use a different underlying web framework to implement the site. As such, they chose to re-implement the components from scratch. This resulted in controls that were superficially similar to the original designs but which behaved quite differently. As a result they required substantial additional refinement to achieve behaviour that was already present in the prototypes.

Often, these issues arise not from limitations in the visualization libraries themselves, but in visualization-adjacent aspects of the implementation such as data loading, cross-browser compatibility, and accessibility requirements. During the development phase of the *Energy Imports & Exports* project, the design team was surprised to learn that the government-mandated accessibility template within which the design would sit also applied to individual components of the visualization. This meant that each selectable element of the visualization would be surrounded by a stroked rectangular box with a dominant color. This late discovery resulted in a mismatch between the visual aesthetic of the design, which was built around hexagonal tiles, and the bright halos produced by the accessibility template (Figure 7). The use of the template also forced the design team to revisit the data mapping to ensure that the color used by the accessibility overlays did not conflict with the palette used to encode import and export data.

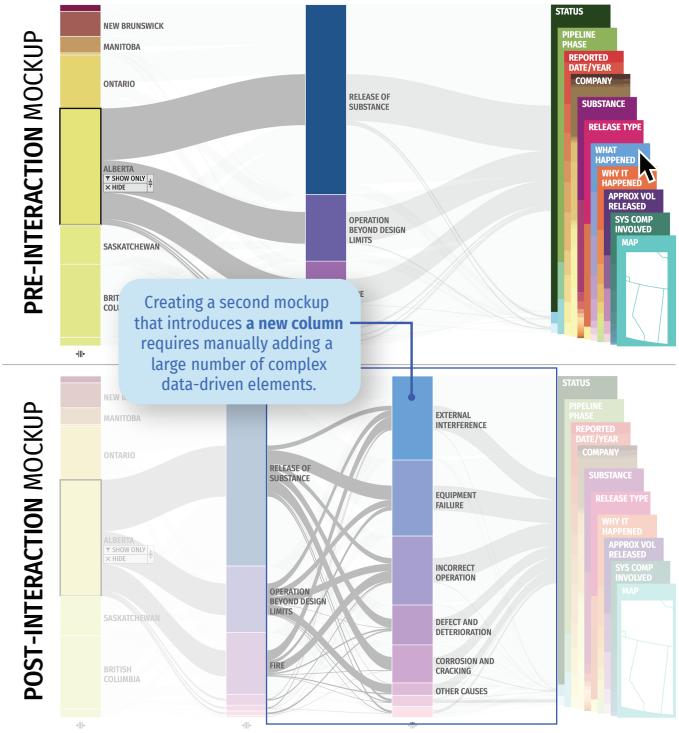

Fig. 8: Two views of a parallel sets-style visualization of incidents on pipelines and pipeline facilities. **Top:** A single category is selected. Highlighted gray lines represent how the incidents in the selected category relate to incidents in the categories of the column to the right. **Bottom:** The design allows additional columns to be dragged into the visualization from the right. This single interaction introduces substantial complexity to the resulting view. Prototyping, testing, and communicating this resulting view in a data-accurate way is difficult using conventional graphic and interaction design tools.

C4. Articulating Data-Dependent Interactions

When ideating and specifying new data-dependent interactions, designers often need to generate a variety of different data-accurate views showing the visualization in multiple states. This extra cost can make data-driven interactions challenging to develop and challenging to other team members.

The outcome of most interactions with a visualization depend on the data itself. Typically, these data-dependent interactions generate new views of the visualization that may represent a different subset of the data, a different transformation of the data, or a different encoding of the data. For instance, filtering operations reduce the set of data being considered. This can result in a change of position or appearance of the remaining elements. Similarly, each brushing and linking requires a change to the encoding applied to a very specific set of marks spread across several views. When prototyping these interactions by hand or using graphic design tools, designers must manually manage and update large numbers of individual data elements. Modeling the transitions between such views is even more difficult, especially when complex animations are required.

Transitions like the one in Figure 8 are particularly challenging. In this example, drawn from our *Pipeline Incidents* project, a simple interaction with the flow visualization can add an additional column, introducing dozens or even hundreds of new arcs. The complexity that this single interaction adds to the view in question is substantial. Multiple forking curves appear with varying positions and thicknesses, all of them related to the previous selection.

Creating a data-accurate version of the resulting view is very difficult using conventional graphic design tools and requires manually

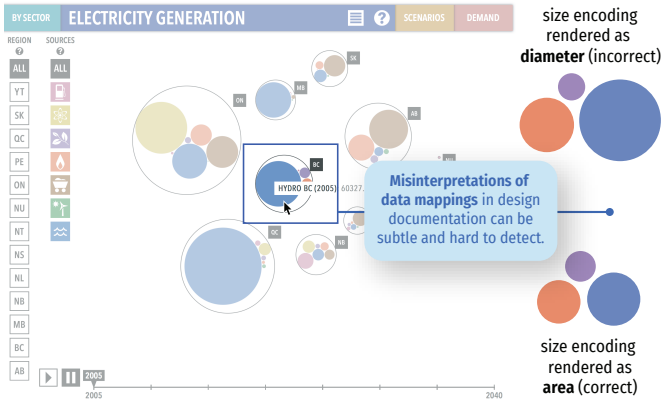

Fig. 9: A static mockup of a bubble chart visualization. While tooltips on the bubbles provide precise values, the image alone does not contain enough information to enable a developer to re-create it. In particular, it does not specify that the data values shown are mapped to the area, not the diameter, of the circles.

computing the size, positions, and connectivity of large numbers of edges, then manually adding them to the mockup. This expense is multiplied every time there is a change to the underlying design or an update to the data. Even trivial changes such as changing screen dimensions or color schemes can require manual revisions to these prototypes. Specifying transitions and animations between these views is also difficult, even when using interactive wireframing and animation tools, since they do not include data binding or animation support for such fine-grained elements. Finally, because this view represents only one of many possible application states, exploring the effectiveness of the interaction requires replicating the process for other views.

Unfortunately, exploring this same interaction by coding a low-fidelity interactive prototype capable of supporting it also entails considerable effort. In comparison to graphic design tools, low-fidelity interactive prototypes also make it considerably more difficult to examine alternative layouts, typefaces, controls, and other aspects of the design. Testing pixel-perfect versions of interactions and transitions in a coded prototype effectively requires committing to and implementing the entire design.

Across the five projects, we often took both approaches, using graphic design tools to sketch and visually polish components and visualization views, while simultaneously implementing prototypes to test the impact of the interaction. We also used interactive charting tools like Tableau [58] and RAWGraphs [40] to create data-accurate visualization elements that could then be exported back into graphic and interaction design tools to create richer mockups. However, the gaps between each of these tools is considerable and interaction prototyping consumed a substantial amount of the design team's bandwidth.

C5. Communicating Data Mappings

Implementing a data mapping correctly requires more detail and precision than can be easily inferred from a mockup of a visualization view. However, precise and complete specification of data mappings is not well-supported by current design tools.

The mapping from data to visual representation is the most fundamental aspect of a visualization. As such, it is critical that a designer creating a visual mapping be able to communicate their intent to others on the team, especially developers. Current options for communicating data mappings are limited and must be created manually. A static rendering of the view may seem sufficient if the visualization is designed to be easily read by non-experts. However example views may not capture many of the nuances and details that are be important for implementation purposes. For example, correctly mapping a data value to a visual mark often requires data transformations or lookups, which may involve multiple hidden steps (such as using a classification algorithm to bin heat map values). Furthermore, the complexity of the data may make it difficult to explicitly show all data cases in the provided views, particularly where interaction is concerned (see Challenge C4–*Interactions*).

One clear example of this issue emerged late in the *Energy Futures* project, which included the bubble chart visualization shown in Figure 9. The visualization design documentation outlining this visualization included one static view of the bubble chart created in Adobe Illustrator [1], together with information about which data columns were to be mapped to the size of the circles. The visualization designers expected, but did not precisely specify, that "size" would be interpreted as area rather than diameter. This expectation originated in their own domain knowledge of common visualization guidelines. However, this guideline is not inherently obvious, particularly to non-experts. In this case, the initial implementation mapped the data to the circles' diameter rather than their area. This subtle difference in the mapping was difficult to detect via visual inspection alone (see Challenge C6–*Integrity*). Ultimately, the error was caught by happenstance. However, once the discrepancy was noticed, extra development time was needed to change the mapping to what the designers originally intended.

Another example from the *Pipeline Conditions* project illustrates that data mapping specification is challenging even in face-to-face conversation. One portion of the visualization displayed a horizontal scrolling list of projects (Figure 10). On either side of this list a bar denoted the number projects to the left and right. At one face-to-face design review session, the issue was raised of how best to map the number of remaining projects to the size of the bars, as the space allotted for the bars is quite small. A simple solution was agreed upon verbally. However, the next implementation cycle revealed that, in fact, even this simple solution could be interpreted in multiple ways: one team understood that the maximum size of the bar would be relative to the highest number of possible projects; the other team understood that the maximum size of the bar would represent a fixed value.

C6. Preserving Data Mapping Integrity across Iterations

Because it is difficult to systematically compare implementations against design documents, there is a serious risk that misinterpretations or misapplications of the data mapping go unnoticed during and after development.

As a design is implemented, differences can emerge due to a variety of factors including bugs, data updates, inconsistencies in the initial designs, and misinterpretations of the data mapping. This is in part because the implementation is separate from the design documentation. The primary method of testing the implementation's adherence to the design is visual inspection and comparison to the original specification. Furthermore, every new iteration of an implementation typically introduces numerous small changes. This can make it difficult to manually keep track of what has and has not been inspected or to know when it is the right time to flag an issue. Even if all parties recognize the importance of preserving the data mapping, visual inspection of the implementation alone may not reveal misinterpretations of the design or the data mapping at any given iteration.

The example in Figure 11 shows a comparison between a small portion of a visualization design document and an implementation of that design. Many small differences related to both presentation design and the data mapping are apparent, including issues with the typography

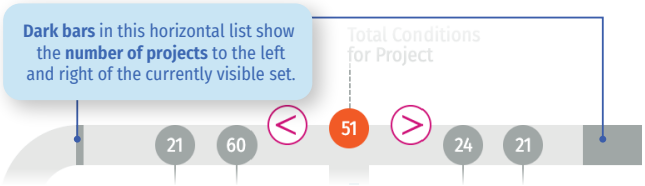

Fig. 10: The mapping between number of projects and the width of the bars in this horizontal list was unclear both in the design document and in follow-up conversations.

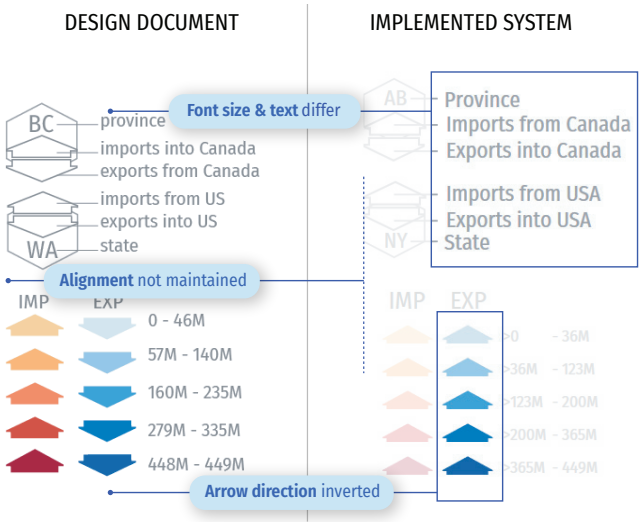

Fig. 11: Detailed differences between a piece of a visualization design as specified in the original design documentation (left) and an implemented version of that design (right). Manually inspecting each iteration of the developed visualization to compare against the design document is tedious and error-prone.

and alignment. In addition, the set of blue arrows in the legend point up, but should point down to align with the downward export arrows in the rest of the visualization.

The incorrectly-sized bubble charts described in Challenge C5 (*Data Mappings*) also illustrate the difficulty of noticing mistakes in implemented designs. While detecting this error using only visual inspection is already difficult, it becomes even more challenging when there are frequent data updates during the visualization development process. Frequent changes and updates like these can lead to uncertainty about whether the visualization does not match expectations due to an incorrect mapping or simply due to differences in the data.

6 Discussion

The challenges we have highlighted stem directly from the intrinsic connection between visualizations and the source data that drives them. When compared against other kinds of interfaces, data affects the visibility of elements, their layout, and their appearance to a much greater extent. Viewers of these tools interact not only with the interface but *with the data*. As a result, developers and software engineers must also deal with the pragmatic limitations of the datasets when considering performance and interactive capabilities. Yet, for designers, anticipating all of the implications of scale and interaction for a given dataset remains challenging. This is exacerbated by the reality that datasets are likely to be updated many times both during and after the design process. Together, these challenges suggest a variety of opportunities for research and tool creation that could specifically support the visualization design process both for individuals and collaborative teams.

6.1 Data Characterization

The process of exploring and characterizing new datasets in preparation for visualization design work has much in common with data analysis and therefore many existing data analysis tools can be appropriated within this space. However, challenge C1 (*Changes*) illustrates some opportunities to create data characterization tools and processes that are dedicated specifically to visualization design. Changes to data characteristics can have a substantial impact upon the final visualization outcome because visualization design choices typically reflect the shape and parameters of the given dataset. However, the extent to which new data may impact the robustness of the visualization may not be clear during the data update process.

Data characterization tools could help remove this disconnect by helping designers better understand how the data has changed from one version to the next and how those changes might alter the design of the visualization. This could include tools for highlighting changes in data column names, extrema, and statistical distributions of data, or for simulating likely future changes based on the current distribution of values. Recent work on semi-automated approaches for outlier detection and profiling in data mining toolboxes like Orange [16] and data discovery tools like DataTours [42] may provide a useful starting point. Similarly, visual tools for quickly comparing the distribution of values in different datasets might help designers more readily detect problematic changes without relying on statistical summaries [38].

6.2 Design Phase

Data-Driven Visualization Ideation. A number of the challenges we experienced are associated fundamentally with the challenge of ideating data-driven visualization designs. Existing commercial tools for manual vector-based graphic design such as Adobe Illustrator [1] have little support for creating complex, data-driven views, while visualization exploration and generation tools such as Tableau [58] or RAWGraphs [40] have limited support for custom visuals and interaction. Meanwhile, programming tools and lower-level libraries can be challenging to use as rapid ideation platforms and can disempower non-programmers. Fortunately, recent projects like Data Illustrator [37], Data Ink [64] and Data-Driven Guides [30] highlight the potential for more expressive data-driven graphic design tools. Several of the challenges we experienced emphasize the need for further work in this space. More direct, dynamic, and expressive tools for designing with data could facilitate rapid exploration of different design alternatives even in the face of changing data (C1–*Changes*). Similarly, more rapid exploration of data-driven design alternatives could make it easier to discover unexpected edge cases (C2–*Edge Cases*) and prototype data-dependent interactions (C4–*Interactions*). Mei et al. [43] identify several additional research directions for these kinds of tools, including supporting refinement based on existing visualizations, providing better debugging support, and exploring programming for dynamic data and interaction.

Where artifact creation cannot be unified into one single authoring platform, better handoff tools may also offer opportunities for designers and developers to synchronize the artifacts across multiple systems. Already, tools like Hanpuku [7] have explored bridging the graphic design expressivity of Adobe Illustrator and the data-driven prototyping capabilities of D3 [10]. However, designing bi-directional workflows between these sorts of existing tools usually entails compromise — often intersecting the limitations of each tool and limiting the pieces of functionality that can be translated. This suggests that unidirectional handoff tools, like those now widely used in interaction design, are a more likely first step.

Data-Driven Interaction Prototyping. Prototyping data-driven interactions within a visualization design is important for exploring different interaction options, ensuring the scalability and understandability of those interaction options, and communicating interaction designs to developers (C4–*Interactions*). Yet data-driven interactions can be complex to prototype because one interaction can simultaneously cause a change to a large number of data-driven elements in a design. Unfortunately, commercially-available user interface prototyping tools do not address this challenge. Commercial interaction design tools make it straightforward to prototype interactions and transitions using static mockups, and even provide some limited support for data-driven prototyping — for example by populating user profiles or lists of information. However, they provide little support for creating visualization views, whose layout, appearance, and interactivity are all deeply and inherently driven by data. As such, opportunities remain for new tools that allow designers to more expressively prototype and test potential interactions either by bootstrapping on top of existing visualization tools or via new authoring interfaces.

Data Mapping Documentation. Communicating and documenting design intent is useful not only for explaining visualization designs to a development team, but also when communicating with other team members or stakeholders and when producing project documentation. One opportunity highlighted by challenge C5 (*Data Mappings*), is to design tools and methods for communicating data mappings. Such

tools would support explicit communication of the relationship between data structures and their graphical representation with enough detail to convey any transformations, calculations, and algorithms required.

Related work on visualization grammars [54, 57, 62] provides a useful starting point, as do projects that represent the visualization pipeline [59] and support the deconstruction and modification of data mappings for existing visualizations [24]. However, there remains a need for data mapping tools that are accessible to designers without programming skills but which still communicate the nuances of a data mapping in enough detail to reproduce it programmatically.

Data Visualization Design Documentation Although data mappings are fundamental to a visualization, they are only one part of its design. Ultimately, any data mapping must be communicated as part of a larger body of design documentation that also captures graphical presentation (including layout, typography, and color) as well as interaction. While currently this documentation is often ad-hoc and informal, more systematic tools for capturing and communicating design details could be valuable in larger visualization design projects. Given the highly visual and interactive nature of visualization designs, one basis for these kinds of documents could be explorable explanations or other interactive documents. This format, popularized by Bret Victor [61] is increasingly used in data analysis practice, and recent literate programming tools like Observable [47], litvis [63], and Idyll [14] may provide promising platforms for creating and documenting visualization designs.

Annotation tools can also play an instrumental role in design communication, particularly when aspects of a design may not be obvious from the visual artifacts alone. Already, annotation is often present in authoring tools for low-fidelity user experience prototypes (cLuster [50], D.Note [26], SketchComm [36], SILK [33]). Going forward, visualization-specific annotation libraries like ChartAccent [51] have the potential to enable richer data-driven selection and annotation that could also support visualization design documents.

6.3 Development Phase

As emphasized by challenge C6 (*Integrity*), it is important that anyone testing an implemented visualization against its design documentation is able to identify discrepancies between the intended design and the implementation. Moreover, they should be able to differentiate between discrepancies that are due to data differences, discrepancies that are due to incomplete implementation, and unintentional discrepancies that are due to miscommunication or mistakes. The most vital discrepancies are those pertaining to data mappings (C5–*Data Mappings*). However, presentation discrepancies related to screen size, alignment, layout, typography, and non-data color issues like those illustrated in Figure 11 also play a role. In these cases, new tools for supporting differencing and visual comparison between visualizations [20, 21] may be particularly valuable. Interaction discrepancies, on the other hand, may be more difficult to detect, but are especially important when they can affect the understanding of the data. Working visualization prototypes that simulate the final interactions can mitigate some of these issues, but only when the technical constraints of the design and development teams are aligned ahead of time (C3–*Constraints*).

6.4 Many Paths to Visualization Design

The challenges for data visualization design and handoff described here are a direct reflection of our own experience in a large, multi-team environment and are limited to our perspective as the design team. With the exception of the first project, the physical and temporal separation between data compilation, visualization design, and development were relatively high. Not all visualization projects are configured in this way. Some projects involve multiple co-located people in highly specialized roles (for example in newsrooms [19], where journalists, data analysts, designers and programmers might work closely together on very tight timelines). Meanwhile, many projects still involve individual designer-developers taking on many roles simultaneously. There is a wealth of literature discussing managing communication on design teams [18], including work on developing shared mental models of tasks, teams, and processes [5, 28], sharing meaning-making activities [34], and managing the organization of design work [13]. As such, it is clear that technology is only one way to help mitigate communication challenges and that processes, communication frameworks, education, environment, mutual trust/comfort, and increased face-to-face time are all factors that affect the design process.

The separation of our teams likely exacerbated and made it easier to identify some of the challenges we articulate. Our projects transitioned from phase to phase in very discrete stages and the need for clear communication and knowledge transfer between teams was very strong. However, the challenges we identify are more closely related to the way that data permeates every aspect of the design process and would remain to some extent regardless of team configuration. As we have shown, data impacts nearly every facet of visualization design. A single update to the data distribution can completely change the effectiveness of the visualization (C1–*Changes*). Incorporating standard data interactions into a design can increase the complexity of the data inputs that need to be supported and make it much more difficult to find edge cases (C2–*Edge Cases*). Moreover, the fundamental dependence of the visualization on data may require a robust software engineering framework, the constraints of which are not always apparent during the design phase, making it difficult to anticipate technical challenges or provide coded prototypes that can easily be translated to final implementations (C3–*Constraints*). Prototyping interactions with data can also involve tedious calculations and significant changes from view to view (C4–*Interactions*). Likewise, articulating and testing the mapping from data to its visual representation is a fundamental and complex task that is not well supported by conventional user interface design tools (C5–*Data Mappings*, C6–*Integrity*). Everything in the visualization design — the mapping, the graphic design, and the interaction design — depends deeply upon the data, and therefore the data adds complexity to every step of this process. Finding and addressing opportunities to support that complexity could make the visualization design process more effective, more expressive, and more accessible to people with a variety of skillsets to offer.

7 Conclusion

Based on reflections on our experiences as designers during the data characterization, visualization design, and development phases of several large-scale collaborative visualization projects, we have highlighted the increased complexity that data fundamentally introduces to the design process. Data-related challenges span all phases of design and include adapting to late-stage data changes, anticipating edge cases, articulating data-dependent interactions, communicating data mappings, and preserving data mapping integrity in implementation. These point to several opportunities to create tools that directly support the visualization design process through specific data-related features. Creating these more powerful tools could make the design process more robust, efficient, and accessible to people in a variety of design roles.

Acknowledgments

We thank (alphabetically) Amanda Harwood, Annette Hester, Ryan Hum, Katherine Murphy, and Peter Watson and their teams at the National Energy Board of Canada (NEB) for facilitating and supporting this project. We also thank the software development team at VizworX. We also thank members of the design teams who took part in the projects which led to this paper (alphabetically) Bon Adriel Aseniero, Peter Buk, Shreya Chopra, Claudio Esperança, Tina Huynh, Lisa Hynes, Lindsay MacDonald Vermeulen, Claudia Maurer, Charles Perin, Ricky Pusch, Lien Quach, Katrina Tabuli, Markus Tessmann, Sarah Storteboom, Sonja Thoma, Jo Vermeulen.

References

[1] Adobe, Inc. Adobe Illustrator. http://adobe.com/products/illustrator/.

[2] Adobe, Inc. Adobe XD. http://adobe.com/products/xd.html.

[3] Atomic.io Limited. Atomic Prototyping. http://www.atomic.io.

[4] Axure Software Solutions, Inc. Axure RP. https://www.axure.com.

[5] P. Badke-Schaub, A. Neumann, K. Lauche, and S. Mohammed. Mental models in design teams: a valid approach to performance in design collaboration? *CoDesign*, 3(1), Mar. 2007. doi: 10.1080/15710880601170768

[6] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Reflections on How Designers Design with Data. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14. ACM, 2014. doi: 10.1145/2598153.2598175

[7] A. Bigelow, S. Drucker, D. Fisher, and M. Meyer. Iterating between Tools to Create and Edit Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1), jan 2017. doi: 10.1109/TVCG.2016.2598609

[8] T. Blascheck, L. M. Vermeulen, J. Vermeulen, C. Perin, W. Willett, T. Ertl, and S. Carpendale. Exploration Strategies for Discovery of Interactivity in Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018. doi: 10.1109/tvcg.2018.2802520

[9] Bohemian B.V. Sketch. http://www.sketchapp.com.

[10] M. Bostock, V. Ogievetsky, and J. Heer. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, dec 2011. doi: 10.1109/tvcg.2011.185

[11] J. M. Brown, G. Lindgaard, and R. Biddle. Joint Implicit Alignment Work of Interaction Designers and Software Developers. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12. ACM, 2012. doi: 10.1145/2399016.2399121

[12] R. V. Bullough Jr and S. Pinnegar. Guidelines for quality in autobiographical forms of self-study research. *Educational researcher*, 30(3):13–21, 2001. doi: 10.3102/0013189X030003013

[13] M.-L. Chiu. An organizational view of design communication in design collaboration. *Design Studies*, 23(2), Mar. 2002. doi: 10.1016/S0142-694X(01)00019-9

[14] M. Conlen and J. Heer. Idyll: A markup language for authoring and publishing interactive articles on the web. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '18. ACM, 2018.

[15] M. Czerwinski, M. Czerwinski, E. Horvitz, and S. Wilhite. A diary study of task switching and interruptions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pp. 175–182. ACM, 2004. doi: 10.1145/985692.985715

[16] J. Demšar, T. Curk, A. Erjavec, Črt Gorup, T. Hočevar, M. Milutinovič, M. Možina, M. Polajnar, M. Toplak, A. Starič, M. Štajdohar, L. Umek, L. Žagar, J. Žbontar, M. Žitnik, and B. Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013.

[17] A. Desjardins and A. Ball. Revealing tensions in autobiographical design in hci. In *Proceedings of the SIGCHI Conference on Designing Interactive Systems*, DIS '18, pp. 753–764. ACM, 2018. doi: 10.1145/3196709.3196781

[18] C. Eckert, A. Maier, and C. McMahon. Communication in design. In J. Clarkson and C. Eckert, eds., *Design process improvement: A review of current practice*. Springer London, 2005. doi: 10.1007/978-1-84628-061-0_10

[19] R. Fischer-Baum and C. Esteban. Working in a graphics visual storytelling team. Keynote presentation, Information Plus Conference, Potsdam, Germany, Oct 2018.

[20] M. Gleicher. Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 24(1), jan 2018. doi: 10.1109/TVCG.2017.2744199

[21] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4), oct 2011. doi: 10.1177/1473871611416549

[22] J. Gray, L. Chambers, and L. Bounegru. *The data journalism handbook: How journalists can use data to improve the news*. O'Reilly Media, Inc., 2012.

[23] H. Hagen, D. Keim, T. Munzner, S. North, and H. Pfister. Vis restructuring recommendations, fall 2018. http://ieeevis.org/governance/1810-Restructuring-Recommendations.pdf, 2018.

[24] J. Harper and M. Agrawala. Deconstructing and Restyling D3 Visualizations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14. ACM, 2014. doi: 10.1145/2642918.2647411

[25] M. Harrower and C. A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.1179/000870403235002042

[26] B. Hartmann, S. Follmer, A. Ricciardi, T. Cardenas, and S. R. Klemmer. D.Note: Revising User Interfaces Through Change Tracking, Annotations, and Alternatives. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10. ACM, 2010. doi: 10.1145/1753326.1753400

[27] InvisionApp, Inc. InVision. http://www.invisionapp.com.

[28] T. E. Johnson, Y. Lee, M. Lee, D. L. O'Connor, M. K. Khalil, and X. Huang. Measuring sharedness of team-related knowledge: Design and validation of a shared mental model instrument. *Human Resource Development International*, 10(4), Dec. 2007. doi: 10.1080/13678860701723802

[29] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, Dec 2012. doi: 10.1109/TVCG.2012.219

[30] N. W. Kim, E. Schweickart, Z. Liu, M. Dontcheva, W. Li, J. Popovic, and H. Pfister. Data-Driven Guides: Supporting Expressive Design for Information Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):491–500, jan 2017. doi: 10.1109/tvcg.2016.2598620

[31] R. M. Kirby and M. Meyer. Visualization collaborations: What works and why. *IEEE Computer Graphics and Applications*, 33(6):82–88, 2013. doi: 10.1109/MCG.2013.101

[32] S. Knudsen, L. Quach, P. Buk, K. Tabuli, S. Chopra, W. Willett, S. Carpendale, J. Vermeulen, D. Kosminsky, J. Walny, M. West, C. Frisson, B. A. Aseniero, L. M. Vermeulen, and C. Perin. Democratizing Open Energy Data for Public Discourse using Visualization. In *Extended Abstracts of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 2018. doi: 10.1145/3170427.3186539

[33] J. A. Landay. SILK: Sketching Interfaces Like Krazy. In *Conference Companion on Human Factors in Computing Systems*, CHI '96. ACM, 1996. doi: 10.1145/257089.257396

[34] A. Larsson. Making sense of collaboration: The challenge of thinking together in global design teams. In *Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '03. ACM, 2003. doi: 10.1145/958160.958184

[35] G. Leiva, N. Maudet, W. Mackay, and M. Beaudouin-Lafon. Enact: Reducing designer–developer breakdowns when prototyping custom interactions. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(3):19, 2019. doi: 10.1145/3310276

[36] G. Li, X. Cao, S. Paolantonio, and F. Tian. SketchComm: A Tool to Support Rich and Flexible Asynchronous Communication of Early Design Ideas. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12. ACM, 2012. doi: 10.1145/2145204.2145261

[37] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data Illustrator: Augmenting Vector Design Tools withLazy Data Binding for Expressive Visualization Authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, 2018.

[38] J. Matejka and G. Fitzmaurice. Same Stats Different Graphs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI'17*. ACM Press, 2017. doi: 10.1145/3025453.3025912

[39] N. Maudet, G. Leiva, M. Beaudouin-Lafon, and W. Mackay. 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*, CSCW '17. ACM, 2017. doi: 10.1145/2998181.2998190

[40] M. Mauri, T. Elli, G. Caviglia, G. Uboldi, and M. Azzi. RAWGraphs: A Visualisation Platform to Create Open Outputs. In *Proceedings of the 12th Biannual Conference on Italian SIGCHI Chapter*, CHItaly '17. ACM, 2017. doi: 10.1145/3125571.3125585

[41] S. McKenna, D. Mazur, J. Agutter, and M. Meyer. Design Activity Framework for Visualization Design. *IEEE Transactions on Visualization and Computer Graphics*, 20(12), Dec 2014. doi: 10.1109/TVCG.2014.2346331

[42] H. Mehta, A. Chalbi, F. Chevalier, and C. Collins. DataTours: A Data Narratives Framework. In *IEEE InfoVis 2017 Posters*, oct 2017.

[43] H. Mei, Y. Ma, Y. Wei, and W. Chen. The design space of construction tools for information visualization: A survey. *Journal of Visual Languages & Computing*, oct 2017. doi: 10.1016/j.jvlc.2017.10.001

[44] M. Meyer and J. Dykes. Reflection on reflection in applied visualization research. *IEEE Computer Graphics and Applications*, 38(6), Nov. 2018. doi: 10.1109/MCG.2018.2874523

[45] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), Nov. 2009. doi: 10.1109/TVCG.2009.111

[46] C. Neustaedter and P. Sengers. Autobiographical design in hci research: designing and learning through use-it-yourself. In *Proceedings of the*

SIGCHI Conference on Designing Interactive Systems, DIS '12, pp. 514–523. ACM, 2012. doi: 10.1145/2317956.2318034

[47] Observable, Inc. Observable. http://www.observablehq.com.

[48] S. Ohly, S. Sonnentag, C. Niessen, and D. Zapf. Diary studies in organizational research. *Journal of Personnel Psychology*, 2010. doi: 10.1027/1866-5888/a000009

[49] F. K. Ozenc, M. Kim, J. Zimmerman, S. Oney, and B. Myers. How to Support Designers in Getting Hold of the Immaterial Material of Software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10. ACM, 2010. doi: 10.1145/1753326.1753707

[50] F. Perteneder, M. Bresler, E.-M. Grossauer, J. Leong, and M. Haller. cLuster: Smart Clustering of Free-Hand Sketches on Large Interactive Surfaces. In *Proceedings of the ACM Symposium on User Interface Software & Technology*, UIST '15. ACM, 2015. doi: 10.1145/2807442.2807455

[51] D. Ren, M. Brehmer, B. Lee, T. Hllerer, and E. K. Choe. ChartAccent: Annotation for data-driven storytelling. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, apr 2017. doi: 10.1109/PACIFICVIS.2017.8031599

[52] M. Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4):21–27, Apr. 1994. doi: 10.1145/175276.175288

[53] J. Rieman. The diary study: A workplace-oriented research tool to guide laboratory efforts. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pp. 321–326. ACM, 1993. doi: 10.1145/169059.169255

[54] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, jan 2017. doi: 10.1109/tvcg.2016.2599030

[55] M. Sedlmair, M. Meyer, and T. Munzner. Design Study Methodology: Reflections from the Trenches and the Stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12), dec 2012. doi: 10.1109/TVCG.2012.213

[56] S. Sonnentag. Work, recovery activities, and individual well-being: A diary study. *Journal of occupational health psychology*, 6(3):196, 2001. doi: 10.1037/1076-8998.6.3.196

[57] C. Stolte and P. Hanrahan. Polaris: a system for query analysis and visualization of multi-dimensional relational databases. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*. IEEE, 2000. doi: 10.1109/infvis.2000.885086

[58] Tableau, Inc. Tableau Software. https://www.tableau.com.

[59] M. Tobiasz, P. Isenberg, and S. Carpendale. Lark: Coordinating Co-located Collaboration with Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1065–1072, nov 2009. doi: 10.1109/tvcg.2009.162

[60] A. Vande Moere and H. Purchase. On the role of design in information visualization. *Information Visualization*, 10(4):356–371, 10 2011. doi: 10.1177/1473871611415996

[61] B. Victor. Humane Representation of Thought: A Trail Map for the 21st Century. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, UIST '14. ACM, 2014. doi: 10.1145/2642918.2642920

[62] L. Wilkinson. The grammar of graphics. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(6):673–677, oct 2010. doi: 10.1002/wics.118

[63] J. Wood, A. Kachkaev, and J. Dykes. Design exposition with literate visualization. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):759–768, 2019. doi: 10.1109/TVCG.2018.2864836

[64] H. Xia, N. H. Riche, F. Chevalier, B. D. Araujo, and D. Wigdor. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '18, 2018.

[65] C. Yiu. Collaboration with distributed teams. *interactions*, 21(4):50–53, jul 2014. doi: 10.1145/2627341