The CAPED Framework: Characterizing User Skills in Chart Authoring through the Lens of Tasks and Tools

Bongshin Lee, Zhicheng Liu, John Thompson, Chenglong Wang

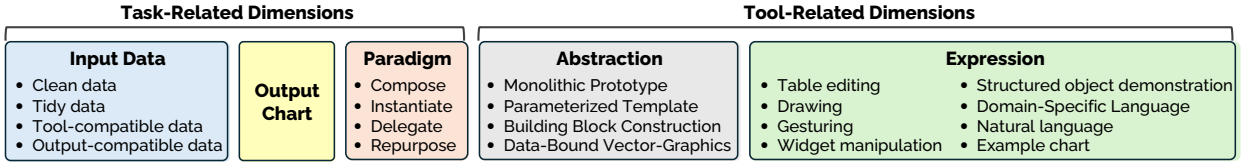


Fig. 1: The CAPED framework enables the characterization of user skills in chart authoring using five dimensions: Input Data, Output Chart, Paradigm, Abstraction, and Expression.

Abstract—In this paper, we introduce a new framework for characterizing the user skills required for chart authoring. Despite growing attention to authoring systems, we lack a conceptual framework that characterizes the diverse and multifaceted skills involved and helps identify opportunities for system design and development. Our initial investigation revealed that skill is a complex and elusive concept, and that existing theories on skills from other disciplines are too high-level to capture the importance of tasks and tools in data visualizations. To address this gap, we formulate the CAPED framework that characterizes skill as the ability to create a chart involving five dimensions pertinent to tasks and tools: input data, output chart, paradigm, abstraction, and expression. We demonstrate the framework's descriptive power by highlighting different skills involved in similar authoring situations and identifying the types of knowledge associated with chart authoring. We further present examples illustrating how the framework can be used to identify new research opportunities in supporting users with varying skills.

Index Terms—Skill, chart authoring

1 Introduction

Creating data visualization is an essential task in both data analysis and presentation, and it has become a cornerstone of data visualization research. However, compared to efforts that characterize and measure visualization literacy for consumption (e.g., [5, 19, 24]), the research community has given little attention to the user skills required for chart authoring. Numerous toolkits, grammars, models, interactive tools, and natural language interfaces have been developed for chart authoring, and many of them boast a reduced learning curve and lower barrier of entry. However, these efforts are system-centric, and it is unclear what user skills are needed to be able to use such tools and interfaces. The creators of interactive authoring tools often simply identify their target audience based on roles (e.g., designers) or implicitly characterize them using terms such as "novices" or "non-experts" [10]. Such terms are ambiguous, often overloaded, and do not identify the skills the users already possess or need to acquire.

More importantly, we lack a conceptual framework to describe and discuss user skills for chart authoring to differentiate diverse, multi-faceted user profiles. For example, one user may be familiar with various types of charts, but has no coding experience; another may be an avid programmer, but has limited knowledge in chart design; a third may be a professional graphic designer, but rarely incorporates structured data into their work. Beyond diverse user skill sets, chart authoring is inherently complex, involving multiple tasks (e.g., data preparation, visual encoding, chart construction and refinement) across tools ranging from programming languages to interactive systems. As a result, different users may tackle the same authoring task using different tools, with different strategies and workflows (Section 2).

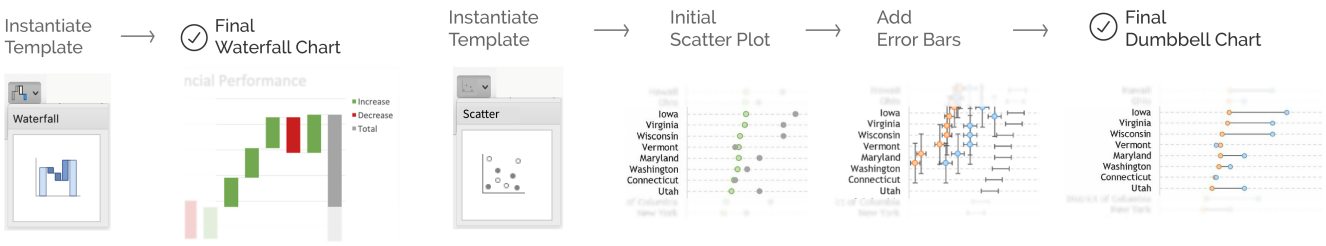
In this work, we aim to have a more systematic way to characterize the user skills required for chart authoring, with the goal of helping the design and evaluation of chart authoring systems. We began by reviewing research on skill from sociology and learning sciences (Section 4.1), which suggests viewing skill as a conceptual construct representing the ability to accomplish a task. We then analyzed chart authoring tasks, but found that focusing on tasks alone was insufficient, as the properties of authoring tools also influence the skills required (Section 4.2). To address this, we examined high-level tool characteristics and synthesized these insights into an integrated account through the lens of authoring tasks and tools (Section 4.3).

The resulting CAPED framework[1] characterizes skill as the ability to create a chart across five core dimensions—input data, output chart, paradigm, abstraction, and expression—pertinent to tasks and tools (Figure 1, Section 5). CAPED is not a taxonomy of skills, nor a process model of chart authoring. Rather, it identifies dimensions that determine what abilities are required in authoring situations. We demonstrate the CAPED framework's descriptive power by highlighting different skills involved in similar authoring situations and identifying the types of knowledge associated with chart authoring (Section 6.1). In addition, we use the framework to identify research opportunities for supporting different users with varying skills (Section 6.2). We conclude with the discussion on the implications and limitations of the CAPED framework, which indicate interesting future research. By providing a unified vocabulary for authoring competencies, the CAPED framework offers a foundational step toward a more human-centered evaluation of visualization tools in an increasingly AI-driven landscape.

- *Bongshin Lee is with the Department of Computational Science and Engineering and the Yonsei Institute of Digital Health, Yonsei University, Seoul, Republic of Korea. E-mail: b.lee@yonsei.ac.kr.*
- *Zhicheng Liu is with the University of Maryland, College Park, MD, USA. E-mail: leozcliu@umd.edu.*
- *John Thompson is with Autodesk Research, GA, USA. E-mail: john.thompson@autodesk.com.*
- *Chenglong Wang is with Microsoft Research, WA, USA. E-mail: chenglong.wang@microsoft.com.*

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

[1] CAPED suggests being "equipped" or having a special skill, akin to being a hero with a cape, symbolizing ability and expertise. The name is composed of one letter from each of the five dimensions: output **C**hart, visualization **A**bstraction, **P**aradigm, **E**xpression, and input **D**ata.



(a) (b)

Fig. 2: An author can instantiate a template to create a waterfall chart in Excel (a), or need non-trivial efforts to "hack" scatter plot and error bar to compose a dumbbell chart (b).

2 Scope and Motivating Scenarios

2.1 What Chart Authoring Entails in Our Framework

Chart authoring corresponds to composing a chart with a design in mind and data in hand to portray specific data variables or insights. It is a necessary step in data visualization for both communication and exploratory analysis. Our framework characterizes the skills required in chart authoring across both contexts. We recognize that chart authoring is an iterative process, where authors typically brainstorm, reflect, critique, and revise a chart until it satisfies their goals. However, we decided not to cover design ideation skills, instead focusing on the more demonstrable action of constructing a chart. In addition, our framework is scoped to chart authoring with abstract data in the information visualization domain.

Authoring typically encompasses all designed aspects of the data experience, this includes the data-to-visual representations, annotations, interactions, and animations. However, as a first step toward developing a vocabulary to describe and discuss authoring skills, we focus on static visual representations and leave other aspects, such as interaction, annotation, and animation, as future work. We also scope our framework to encapsulate only digital tools. While hand-drawn [27], painted, or fabricated visualizations are expressive mediums for representing data, this practice has significantly different backgrounds and skills compared to authoring with graphical user interfaces (GUIs). In addition, our framework scopes data processing skills based on whether or not the end goal is to prepare the data for composing a chart. We do not include statistical data analysis skills that involve methods, tools, and techniques to analyze data in the absence of charts.

2.2 Example Scenarios Requiring Diverse Skills

Chart authoring involves several different activities, including data preparation, chart construction, and chart refinement, as well as a broad range of tools from programming libraries to interactive tools. One person may have different levels of expertise for each of these activities and associated tools. In this section, we describe three example scenarios to showcase the diverse required skills in visualization authoring. We refer to these three scenarios throughout the paper as running examples. To complement the descriptions in this section, the chart images together with the full scenario text are available on our supplemental website: `https://caped-framework.github.io/`.

Scenario 1: Two Different Charts with the Same Tool

Alex is an analyst who is asked to create a waterfall and dumbbell chart for a quarterly financial performance report. Alex decides to use Excel, a tool they are familiar with, to create these charts. A waterfall chart will illustrate changes in the company's net income per quarter. Alex prepares the data to conform to Excel's waterfall chart template: grouping by quarter and computing the difference from the previous quarter. With the data properly formatted, Alex produces the chart by selecting "Waterfall" chart from the "Insert" tab in Excel (Figure 2a).

A dumbbell chart will show the company's actual performance against its targets. Excel lacks a dumbbell chart template, so Alex searches online and decides to follow a tutorial to create a dumbbell chart with work-arounds[2] (Figure 2b). Following the tutorial, Alex first organizes the data to "Insert" a "Scatter Plot". As a work-around, Alex then inserts error bars to connect the target and actual points. Finally, Alex formats the chart to remove unwanted visuals from the error bars to achieve a dumbbell chart.

Remarks. Given that Alex can fluently use Excel templates to create the desired waterfall chart, a non-trivial chart, one might consider Alex is *skilled* in chart authoring with Excel. However, Alex's struggle with dumbbell chart authoring contradicts this assessment. In fact, such struggle does not mean they are *unskillful* with Excel either, since the two chart creation processes employ completely different authoring approaches: the first involves instantiation of built-in templates from the tool, while the latter requires "hacking" to appropriate other plots to mimic the dumbbell chart. This scenario highlights that chart authoring skills *cannot be characterized solely at the level of the tool*. Even with the same tool, different tasks may require different authoring approaches and, consequently, different user skills.

Scenario 2: Skills Transfer between Tools

Kori is a data engineer at a public health organization. Kori is asked to create a report that explains the leading causes of death for citizens from 1970 until the present year. The dataset includes mortality rates based on the cause of death (e.g., Cancer, Heart Disease, AIDS) for each year, age group, state residence, and gender. Kori has a programming background and extensive experience creating data visualizations using libraries, however, Kori's organization exclusively publishes with Microsoft Power BI.

Kori decides to visualize the change in mortality rates across age groups. They create a multi-series line chart of "Age Group" displaying "Year" on the x-axis and "Mortality Rate" on the y-axis. Noticing a steady yet gradual decline in all age groups, Kori wonders if the orders of magnitude of the line chart are overshadowed by the older age groups. Kori hypothesizes that a rate of change would allow better comparison. For this chart, Kori needs a new column of "Mortality rate change (since 1970)". They opt to write Python script to transform the data. Using the Pandas data library, Kori loads the dataset, computes the new column, and exports the resulting dataset. After loading the dataset (with all the required fields) back into Power BI, Kori uses drag-and-drop interactions to create a line chart that shows on the whole, all age groups exhibit similar trends of decrease, except for a noticeable peak for the "25 to 44" age group that coincides with the AIDS epidemic.

Remarks. Despite being relatively new to Power BI, Kori quickly adapts to it based on experience from declarative programming libraries (e.g., Altair). Chart authoring skills are transferable, beyond the authoring medium (in this case Kori was more familiar with textual code rather than a shelf-construction UI), as Altair and Power BI share similar authoring concepts of mapping data columns to visual channels. However, transfer of skills is not bi-directional as someone with only Power BI experience would need to learn additional programming skills to use Altair. Thus, when characterizing skills, we need to consider not only data visualization concepts but also the medium to express

[2] https://simplexct.com/how-to-create-a-dumbbell-chart-in-excel

these concepts. This scenario also highlights the importance of data transformation in the chart authoring process. While this step is often overlooked, it becomes essential when authors need to test hypotheses and prepare the desired charts. In Kori's case, they relied on a familiar programming paradigm to transform data. Alternatively, they could have used Power Query (a companion tool of Power BI) to perform the transforms without incurring interoperability costs.

Scenario 3: Different Authoring Skills to Work with AI

Jesse, an analyst experienced in both programming and visualization design, wants to compare four different job offers based on: Annual Salary ($), Commute Time (minutes), Vacation Days, Remote Work, and Retirement Match. Jesse uploads the dataset to an AI assistant and prompts: "Create a bar chart to compare these job offers." The AI assistant defaults to generating Python code for a standard grouped bar chart, directly comparing these values side by side. However, Jesse notices only one bar (Annual Salary) is shown for each offer. Jesse examines the code generated by the AI assistant, and confirms that the code is correct but still cannot see the other missing data. Realizing these five metrics span markedly different value ranges–many values are not visible to the eye–Jesse concludes a grouped bar chart is inappropriate in this context.

Jesse now provides a specific follow-up prompt to avoid ambiguous interpretation from the AI assistant: "Change this to a parallel coordinates plot. Give each variable its own vertical axis, normalize the scales, and color the lines based on the job offer." The AI assistant handles the complex data normalization and plotting logic, and successfully generates a readable parallel coordinates plot. Jesse can now compare the jobs across all dimensions.

Remarks. With LLMs, Jesse can generate complex data visualizations like a parallel coordinates plot. Thus, it is tempting to assume that the AI assistant "removes" the chart authoring skill requirement entirely. However, Jesse's success does not come from the AI agent; it relies on a combination of programming and visualization composition knowledge. While the AI accelerated the generation of complex code, Jesse's programming expertise was essential to first inspect and confirm code was syntactically correct but inappropriate in terms of chart design. Jesse then needed the conceptual knowledge to diagnose why the grouped bar chart failed and the precise visualization vocabulary to instruct the AI on the appropriate fix. Automated tools do not eliminate traditional skills; rather, they transform them. Chart authoring skills must evolve to encompass code verification, targeted prompting, structural critique, and the ability to steer AI agents away from conceptually flawed defaults and toward appropriate solutions.

3 Background and Related Work

This section briefly reviews prior work related to chart authoring skills, including system-centric perspectives, user characterization, task taxonomies, and visualization literacy.

System-Centric Perspectives of Chart Authoring Systems. Many chart authoring systems are developed to lower the barrier for non-technical users (or "novices"), and they are often evaluated through user studies using traditional performance metrics such as task completion time and success rate. However, these studies typically focus on usability and do not enable systematic comparisons across authoring tools with similar goals. Furthermore, they offer limited insights into the process, required skills, and cognitive aspects of authoring tasks [34]. The critical reflections method [36] was proposed to address some of these limitations by enabling in-depth analysis and comparison of the components and authoring processes in multiple systems. Nevertheless, it remains system-centric, is better suited for assessing a small number of authoring tools, and does not generalize to systems that embody diverse authoring paradigms and expressions. It thus falls short of supporting systematic discussions or guidelines covering diverse tools from a user's perspective. Our CAPED framework provides a starting point for a user- and skill-centered characterization of chart authoring.

Target User Characterization. *Novices* have been popular target audience for data visualization: many visualization research projects often use terms such as novices and non-experts to refer to target users with little experience or skills in chart authoring [10]. For example, Elias and Bezerianos' work [14] reports the design and evaluation of Exploration Views, a system that allows novice visualization users to easily build and customize Business Intelligence information dashboards, and Grammel et al.'s work [16] uses the term novices to characterize people with little experience or skills.

Burns et al. recently investigated how visualization researchers define novices and evaluate visualizations intended for them, revealing a substantial gap between the broad language used to describe novices and the narrow population representing them in evaluations (i.e., young people, students, and US residents) [10]. The term "non-experts" has similar issues and lacks precision. Some chart authoring research identifies job roles (e.g., graphic designers) when characterizing their target audience, but they still under-specify the backgrounds and skills of people in the same profession. The CAPED framework provides a vocabulary to more precisely describe user skills along the five proposed dimensions, highlighting the fact that user characterization requires more nuanced approaches.

Visualization Task Characterization. Visualization task taxonomies emerged from the need to design and evaluate visualizations from a user-centered perspective. Early work focused on relating tasks to data and interaction, such as Shneiderman's "task by data type" taxonomy [40], Valiati et al.'s low-level tasks for multidimensional visualization [44], and Amar et al.'s taxonomy of low-level analytic tasks [3]. These frameworks helped describe what users can do with visualizations and supported empirical comparisons between techniques. Later work introduced higher-level views of visualization design. Tory and Möller framed visualizations as data transformation processes [42], while Munzner's nested model framed domain problems, data and task abstraction, visualization encoding, and visualization algorithms as tiered steps in the design process [31]. Brehmer and Munzner further bridged levels of task granularity through their multi-level typology [9], and these ideas were synthesized in Munzner's broader framework for visualization analysis and design [32].

More recent work has shifted from describing visualization tasks to describing how people author visualizations [20, 30, 48]. Authoring paradigms on visualization construction [20] and bottom-up versus top-down authoring [30] are especially relevant. Our work builds on this direction by characterizing chart authoring in finer detail through five core dimensions (Figure 1, Section 5).

Assessment of Literacy or Ability. Visualization literacy involves a person's ability to read, understand, and use visualizations to answer questions [5]. We contend that literacy is a prerequisite to authoring a visualization. Boy et al. [8] and Lee et al. [24] introduced the first literacy tests to quantitatively evaluate a user's comprehension abilities. More recently, research on visualization literacy has evaluated visual encoding ability [15], pushing the definition of literacy borrowed from textual literacy to one based on skillsets [41]. While our approach does not provide a metric for skills, it deconstructs visualizations based on data, abstractions, and paradigms. As we discuss in Section 7, we believe our framework could support the development of authoring skill assessments similar to literacy tests. We also envision that a deeper understanding of chart authoring skills will help advance AI-assisted chart authoring. For example, Hong et al. took a crucial first step by evaluating the visualization literacy of LLMs [19]. Building on our framework, we identify new research opportunities at the intersection of AI and chart authoring in Section 6.2.

4 Framework Development and Skill Framing

Motivated by the lack of a theoretical framework to describe and discuss chart authoring skills, we adopted an iterative conceptual synthesis approach involving weekly discussions among the four authors over a 16-month period. We began by reviewing existing theories of human skill and competence from other fields and examining their applicability to chart authoring. This review suggested construing skill as the ability to successfully accomplish a task (Section 4.1). Guided by this perspective, we next analyzed common activities and tasks involved in transforming input data to output charts. Through this analysis, we

found that task descriptions alone were insufficient to characterize chart authoring skills (Section 4.2). We therefore conducted a comparative analysis of various chart authoring tools, identifying four recurring authoring approaches and abstracting common properties across tools. Finally, we synthesized these insights into a unified framework organized around the lens of tasks and tools (Section 4.3).

During this process, our focus emerged on viewing skill as a conceptual construct rather than a measure of proficiency (e.g., novice versus expert). The resulting framework is not a taxonomy of skills; rather, it identifies dimensions that determine what skills are required in a chart authoring situation. Consequently, the dimensions in the framework should not be interpreted as skills themselves. For example, a desired output chart is not a skill; rather, a specific output chart may require knowledge and abilities such as understanding the structure of the target chart and determining whether it can be created using a given tool. We also note that our work does not address levels of skill or expertise.

4.1 Importance of Tasks in Skill Characterization

Multiple disciplines, including psychology, sociology, and economics, have tried to define and characterize the concept of "skill" [4, 17] or "competence" [23]. Many view skill as the ability to accomplish a task, and the level of skill is associated with the complexity of the task: "*to exercise greater skill is to carry out a more complex activity*" [4]. Similarly, competence refers to "*the ability to successfully perform a range of tasks to a high level of performance*" [17]. While economists tend to view skill and competence as interchangeable concepts, psychologists view skill "*in a narrower way to whether some can do some task or set of tasks*" and define competence as a broader concept [23]. This broader view of competence identifies four dimensions [23]: *cognitive* (know-that and know-why, which include both conceptual and tacit knowledge), *functional* (skills or know-how, things that a person should be able to do or demonstrate), *social* (know-how-to-behave, possession of appropriate values and ability to make sound judgments), and *meta* (ability to acquire the previous three competencies, cope with uncertainty, and learn and reflect).

These four dimensions bear resemblance to theoretical frameworks from education and learning science. For example, Bloom et al. outline three domains of learning: *cognitive* (intellectual skills related to knowledge and problem solving), *affective* (attitudes, values, and interests), and *psychomotor* (ability to physically accomplish tasks) [7]. Krathwohl further revised the taxonomy in the cognitive domain to identify four dimensions of knowledge and six dimensions of cognitive processes. The four dimensions of knowledge are *factual* (e.g., terminology), *conceptual* (e.g., categories, generalizations, and models), *procedural* (e.g., procedures and algorithms), and *metacognitive* (e.g., strategies and self-knowledge). The six dimensions of cognitive process are *remember*, *understand*, *apply*, *analyze*, *evaluate*, and *create* [21].

Two major themes emerge from these bodies of work. First, the notion of skill is closely associated with the ability to accomplish a task. Second, skills can be expressed as the possession of different types of knowledge. In the context of this paper, we can thus define chart authoring skills as the ability to create specific charts with given datasets. Such skills entail the acquisition of knowledge such as conceptual knowledge (e.g., the meaning of mark and axis), procedural knowledge (e.g., how to compose a chart using a specific tool), and metacognitive knowledge (e.g., how to choose a different construction strategy if the current one fails).

This definition, however, is too high-level to be useful in identifying and discussing the skills needed for chart authoring. Enumerating different types of knowledge required for an authoring task is practically impossible. Kusterer calls this the "infinite regress problem": even for the most mundane skill, it is impossible to write down everything that one needs to know [22]. On the other hand, while task is a well-studied topic in the data visualization literature, most efforts focus on tasks involved in *using* visualizations for data analysis, with few attempts to formulate chart authoring tasks. To address this gap, we decided to explore characterizing chart authoring tasks by identifying subtasks at multiple levels of granularity across different stages in the authoring process as described in the next subsection.

4.2 Analysis of Tasks and the Indispensability of Tools

In our attempt to characterize chart authoring skills, we tried to develop a taxonomy of tasks. Inspired by Brehmer and Munzner's approach [9], we focused on the *multi-faceted* and *multi-level* nature of chart authoring tasks. A task is multi-faceted in the sense that it can be described along three dimensions: *why* the task is performed, *how* the task is performed, and *what* inputs the task takes and outputs it produces. A task can also be described at multiple levels of abstraction, e.g., high-level (consume) to mid-level (search) to low-level (query) for the *why* dimension [9].

Similar to Brehmer and Munzner, we started by formulating abstract authoring tasks that were independent of the properties and designs of specific tools. We observed that many authoring tasks could be grouped into a small number of recurring high-level approaches. For example, some tools require authors to compose charts from primitives, while others allow authors to instantiate predefined templates or examples. These observations led to the identification of recurring authoring paradigms that later became one of the dimensions in the framework. In addition, many common tasks such as data transformation and visual mapping, as identified in the InfoVis Reference Model [11], depend heavily on the characteristics of input data and output chart. For example, the specific data transformations required depend on both the input data format and the output chart structure. Instead of enumerating all such individual tasks in the framework, we chose to incorporate input data and output chart as framework dimensions, allowing the framework to capture the underlying factors determining skill requirements in a more concise way.

Once we began describing chart authoring tasks at the lower levels, however, we realized that it is difficult to do so without referring to the properties, designs, and capabilities of the authoring tool to be used. This observation challenged common wisdom and approach in skill characterization and visualization task abstraction. For instance, none of the theories from psychology and social sciences we reviewed (Section 4.1) tried to incorporate tools as a critical part of the formulation of skills or competence. Existing frameworks of visualization tasks also argue that the specific features of a tool may not be a top priority. When theorizing about the role of interaction in data visualization, for example, Yi et al. observed that "for different representation techniques, different interaction techniques are used to perform a similar task or achieve a similar goal," and thus it made sense to focus on "what users achieve by using the interaction techniques rather than how the techniques provided by Infovis systems work" [53]. Brehmer and Munzner also argued that a task typology that abstracts away the tool features provides "a consistent lexicon for description that supports making precise comparisons of tasks between different visualization tools and across application domains" [9]. The *how* part of their typology thus focuses on abstract methods associated with interaction techniques to encode data, and to manipulate or introduce visual elements.

This line of approach makes sense when the goal is to achieve a tool-agnostic and domain-independent understanding of visualization tasks. However, for our purpose of characterizing user skills in chart authoring, it is not meaningful to focus on abstract tasks alone without considering tools. For example, when authoring a multi-series line graph (Scenario 2) using two different tools (Altair and Power BI) with the same input dataset, the high-level task (compose) and the lower-level tasks (e.g., import data, specify encodings) can all be the same, but different tools entail different skill requirements. Some skills are transferable across tools (e.g., knowledge of the concept of encoding), whereas others are tool-dependent (e.g., data binding via programming vs. GUI). In addition, the same tool can also be used in different ways to create the same chart, involving different tasks and entailing different skills. For example, users can use Charticulator [33] to create a ribbon chart either from a predefined template or from scratch. The former expects authors to prepare the input data in a template-compatible format while the latter requires them to *compose* a chart in addition to preparing the data in a tool-compatible format. These observations suggested that characterizing chart authoring skills requires considering not only the tasks being performed but also the tools through which those tasks are accomplished.

4.3 Analysis of Tools and Synthesizing CAPED

Given the diversity and variety of chart authoring tools, and the fact that existing authoring tools keep evolving and new tools emerging, it is not feasible to enumerate all the possible tools and their features. We thus searched for higher-level concepts that remain stable across tools, and integrate the accounts on tasks and tools in a unified framework.

We initially considered the following dimensions for characterizing the various aspects of chart authoring tools: system model (how charts are internally represented and updated in a tool), user interface (the means through which users operate the tool and receive feedback on their actions), modality (the input/output channel between the user and the authoring tool), device (the hardware device with which users perform chart authoring), and rendering (how a tool renders a chart, e.g., raster or vector graphics). We then tried to apply these dimensions to analyze the authoring experiences in various tools, which made us realize several limitations in focusing on these dimensions.

While system models are important in understanding how a tool works, relevant details are not always available, especially in commercial tools. Furthermore, it is usually more important for users to understand the tool's vocabulary or conceptual building blocks than its internal representations. For example, users do not have to understand VizQL to construct visualizations in Tableau; they just need to have conceptual knowledge of visual encodings that specify the mappings between the data attributes and the visual channels. On the other hand, while UI, modality, device, and rendering can capture important tool properties from a human-computer interaction perspective, they are insufficient in capturing tool properties specific to data visualization. For example, we cannot differentiate the skills required by programming approaches to chart authoring such as D3 and Vega-Lite because they share the UI (code editor), modality (mouse and keyboard), device (desktop), and rendering (SVG). Throughout this process, two dimensions emerged as more appropriate than the aforementioned dimensions: the conceptual abstraction exposed by a tool (e.g., monolithic prototype, parameterized template, building block construction, or data-bound vector-graphics) and the means through which users express intent (e.g., natural language, widget manipulation, table editing, drawing, or programming languages).

Finally, we synthesized these insights into a unified framework organized through the lens of tasks and tools. The resulting framework was iteratively refined through the analysis of representative authoring scenarios, comparisons across chart authoring tools, and discussions of skill transfer between authoring environments.

5 The CAPED Framework: Authoring Skills through the Lens of Tasks and Tools

The CAPED framework consists of five dimensions that determine the knowledge and abilities required in a chart authoring scenario: output **C**hart, chart **A**bstraction, **P**aradigm, **E**xpression, and input **D**ata. The acronym *CAPED* evokes the idea of being "equipped" with the skills and expertise needed for chart authoring, much like a hero wearing a cape. The five dimensions are described below:

- *Input data* (task - what): the dataset to be visualized, which may be clean, tidy, tool-compatible, or template-compatible. As explained earlier, these different types of datasets imply different levels of skills. For example, preparing a tool-compatible dataset requires the conceptual knowledge of the format assumed by a chosen authoring tool as well as the ability to transform a dataset into the required format.
- *Output chart* (task - what): the target chart design to be created. Authoring a particular chart usually requires the ability to describe, read, and understand the chart.
- *Paradigm* (task - how): the high-level approach to author a chart. Example paradigms include composition, instantiation, delegation, and repurposing. The paradigm is closely associated with the authoring tool. For example, D3 users tend to compose charts. However, the same tool may support multiple paradigms, and the same output chart may be created using different paradigms in the same tool.
- *Abstraction* (tool - what): the conceptual framework of what constitutes a chart. Example abstractions include parameterized templates, building block construction, data-bound vector-graphics, and monolithic prototype.
- *Expression* (tool - how): the medium through which users articulate their intents. Expressions are closely related to abstractions. Example expressions include table editing, programming in DSL, UI widget manipulation, and natural language utterances.

5.1 Task-Related Dimensions: Input, Output, Paradigm

The characterization of chart authoring tasks focuses on the *what* and *how* aspects. The *why* dimension is straightforward with a single objective at the top level: to author a chart. At the next level, we identify two tasks that correspond to the underlying reasons (or contexts) for authoring: understand data during the exploratory data analysis and present the data or insights for the communication purposes. We thus elaborate on the what and how dimensions.

5.1.1 What: Input Data and Output Chart

We identify two types of input for a chart authoring task at the top level for the *what* dimension: (1) dataset and (2) a chart template or example required by some tools (e.g., Chartreuse [13] and Mystique [12]). The output of the authoring task is a functional chart that encodes the input data. At a lower level, the input dataset may be characterized as:

- *Clean* data: dataset without errors, missing values, and redundancies, and thus ready to be used as is.
- *Tidy* data: clean data organized into a consistent structure [50].
- *Tool-compatible* data: dataset formatted to be compatible with the target tool. For example, Tableau [1] expects the data table to be in the long format [50], so that attributes can be dynamically mapped to visual channels.
- *Output-compatible* data: many tools not only require the dataset to be in a specific format (e.g., long or wide), but also assume that the dataset conforms to a particular schema that is compatible with the output. For example, Mystique [12] analyzes an example SVG chart to infer the required number of categorical and quantitative columns, and performs a compatibility check on the user's data.

These dataset categories are useful when dealing with authoring tools that require different skills in data transformation. For instance, tools like Falx [45] and Data Formulator [47] require the input data to be tidy, alleviating the burden of performing complex data transformations (e.g., pivoting), while many authoring tools require the input data to be formatted in a tool-compatible manner [36].

5.1.2 How: Authoring Paradigm

For the *how* dimension, we identify the following four tasks, each describing an authoring paradigm characterized by the amount of available scaffolding that embodies the desired chart output.

- *Compose:* authoring a chart by composing a set of predefined primitives or building blocks from scratch. Examples include composing a radial network diagram with hierarchical edge bundling using D3's cluster layout and radial line generator, or building a multi-series line chart by dragging and dropping data attributes to the construction shelf in Power BI (Scenario 2). In both cases, tool-compatible datasets are required.
- *Instantiate:* authoring a chart by infusing new data into a template or chart example supported by a tool. Examples include creating a waterfall chart using Excel's built-in template (Scenario 1), or creating a diverging stacked bar chart using Mystique [12] by providing an SVG example. In both of these cases, tool-compatible datasets are required.

- *Delegate:* authoring a chart by delegating the task to a human or a software agent. Examples include generating a parallel coordinates plot by prompting ChatGPT (Scenario 3), or creating a stacked area chart through Tableau's Show Me recommender [28]. Tidy data is required as input in the former case, while tool-compatible data is required in the latter case.
- *Repurpose:* modifying the components (e.g., mark, layout, encoding) of an existing chart to author the desired chart output. Examples include creating a dumbbell chart by modifying a scatter plot in Excel (Scenario 1). In this case, tool-compatible input data is still required.

We want to emphasize that these four tasks are not meant to be exhaustive. They instead represent common high-level authoring approaches observed in research and practice. Furthermore, accomplishing an authoring goal may require a series of these four authoring tasks. In Scenario 1, for example, Alex needs to first *instantiate* a scatter plot, and then *repurpose* it to create a dumbbell chart.

These tasks can be complemented by low-level descriptions of the actions and operations involved. For example, to compose a multi-series line graph in Scenario 2, Kori needs to first *transform* data to prepare the input data, then to *import* the data into the authoring tool (Power BI), and finally to *configure* the construction shelf. Each of these actions may be decomposed into even lower-level operations. For example, configuring the construction shelf involves *operations* such as *dragging and dropping* columns to the corresponding shelves.

5.2 Tool-Related Dimensions: Abstraction and Expression

Chart authoring tools can be characterized along two key dimensions: Abstraction and Expression.

5.2.1 Abstraction

Chart abstractions are the conceptualizations of charts exposed to the user through a tool's interface. We identified the following types of chart abstractions:

- *Monolithic Prototype*: in this abstraction, a chart is conceptualized using chart typology. For example, a chart created using Excel's template is associated with a chart type name and a representative icon. Customization is primarily limited to styling and reference elements like axis and legend.
- *Parameterized Template*: many tools employ a top-down approach to chart authoring, where users start with a chart template and fill in the data attributes to be mapped to various visual channels. For example, the Ivy system [29] allows users to select from a collection of parameterized templates, and then set the parameters by manipulating widgets or editing a textual specification.
- *Building Block Construction*: in this approach, the abstraction is the product of compiling a specification, which is often declarative. This kind of abstraction is found in libraries based on the Grammar of Graphics [51] such as Vega-Lite. It also includes authoring tools built on top of such specifications such as Lyra [35]. Users create data visualizations in a bottom-up approach by assembling chart building blocks (e.g., mark, encoding). Modifying the chart involves changing the specification at the level of the building blocks, either through a programming API or a GUI.
- *Data-Bound Vector-Graphics*: a chart can also be viewed as a vector graphic in which certain visual properties are determined by data. The incorporation of data binding can be achieved either through a programming API (e.g., D3.js) or a GUI (e.g., Data Illustrator [26]).

It is important to note that the same type of abstraction can be used in different authoring paradigms. For example, both the instantiate and delegate paradigms can treat an output chart as a monolithic prototype. Conversely, the same authoring paradigm may be realized based on different abstractions. To compose a chart, for example, one can choose a GoG library (building blocks specifications) or D3.js (data-bound vector-graphics); Charticulator allows users to create charts through either building block construction or parametrized templates.

5.2.2 Expression

Expression is the medium through which users articulate their intents when authoring a chart. While abstraction is about *what* kind of conceptual descriptions underlie a chart, expression is about *how* that conceptual description is articulated by users in the tool. Expression implicitly captures the corresponding user interface, modality, and device, but places a stronger emphasis on user action and the target object of the action. We identify the following types of expressions:

- *Table editing*: in tools like Excel, users create charts through static templates. Users edit data in a spreadsheet interface so that the data format and schema are compatible with the chosen template.
- *Drawing*: users can "sketch" partial or complete data visualizations with a digital pen or a mouse to convey the desired design. The drawing can be casual and low-fidelity (e.g., Data Ink [52]), or high-fidelity with precise control over the details (e.g., using professional tools like Adobe Illustrator).
- *Gesturing*: users express their intents through mouse, touch (e.g., on touch screens) or spatial gestures (e.g., in VR environments).
- *Structured object demonstration*: users edit chart components by example to illustrate and extrapolate to the whole chart (e.g., Falx [45], StructGraphics [43])
- *Widget manipulation*: users manipulate widgets to control visual properties or shelf configuration (e.g., Tableau [1]).
- *Domain-Specific Language (DSL)*: users employ declarative (e.g., Vega-Lite [37], ggplot2 [49]) and imperative (e.g., Mascot [25], PICCL [39]) programming languages.
- *Natural language*: users convey their intents in natural language (e.g., conversational agent), using either as typed text or as speech-to-text input.
- *Example chart*: in some tools, users can convey the target chart by providing an example in a tool-compatible format. For example, Mystique [12] expects an example chart in the SVG format, while Charticulator [33] requires the chart template to be a tool-compatible JSON file, containing information necessary to render a chart with a template-compatible input data.

The same expression may be used with different chart abstractions. For example, widgets can be used on both monolithic prototypes and parameterized templates. Conversely, the same abstraction may be articulated using different expressions: building block specification may be done using widget manipulation, natural language, or DSL.

6 Applying the Framework

6.1 Using the Framework to Describe

In this section, we demonstrate how to apply the CAPED framework to describe skills required in different authoring situations, and use the framework to describe skills as knowledge. This section refers to the scenarios explained in Section 2.2. To follow along with the chart examples, please visit our supplemental website: `https://caped-framework.github.io/`.

6.1.1 Describing Authoring Scenarios

Authoring different charts using the same tool. Table 1 summarizes different authoring activities using the same tool, including those described in Scenario 1 (Excel) and Scenario 3 (ChatGPT). We use $\rightarrow$ to denote the transition between different stages of authoring that involve different paradigms, and + to denote the combined use of multiple abstractions or expressions during a stage. For example, in Scenario 3, Jesse delegates the authoring of two different charts to an AI assistant. Depending on the outcome, different abstractions and expressions are involved. When AI generates the desired output, the chart abstraction can stay at the level of monolithic prototype; however, once something goes wrong, Jesse needs to dive into the abstraction level of building block construction to inspect the code produced by AI in a domain-specific language.

Tool	Input Data	Output Chart	Paradigm	Abstraction	Expression
Excel	template-compatible data	waterfall chart	instantiate	monolithic prototype	widget manipulation + table editing
	template-compatible data	dumbbell chart	instantiate → repurpose	monolithic prototype → data-bound vector-graphics	widget manipulation + table editing
ChatGPT	clean data	grouped bar chart	delegate → compose	monolithic prototype → building block construction	natural language + DSL
	clean data	parallel coordinates plot	delegate	monolithic prototype	natural language

Table 1: Two examples of authoring different charts using the same tool, Excel (Scenario 1) and ChatGPT (Scenario 3).

Output Chart	Input Data	Tool	Paradigm	Abstraction	Expression
multi-series line chart	tool-compatible data	Altair	compose	building block construction	declarative DSL
	tool-compatible data	PowerBI	instantiate	parameterized template	widget manipulation
dumbbell chart	clean data	Data Formulator	instantiate + delegate	parameterized template	natural language, widget manipulation, DSL
	clean data	Falx	delegate + instantiate	building block construction	structured object demonstration, widget manipulation

Table 2: Two examples of authoring the same chart, multi-series line chart (Scenario 2) and dumbbell chart, using different tools.

Authoring the same chart using different tools. In Scenario 2, Kori can author the same multi-series line graph using Altair or Power BI, as shown in Table 2. The demonstrative differences in skill are the paradigm (compose vs. instantiate), abstraction (buidling block construction vs. parameterized template) and expression (a declarative DSL vs. widget manipulation). We hypothesize that skill transfer between Altair and Power BI requires relatively low effort compared to other skill leaps between two tools. However, skill transfer is not symmetric: widgets are more familiar to users than DSLs.

In other cases, two tools might appear similar at first glance (e.g., Data Formulator [47] and Falx [45]), but their abstractions require users to approach authoring with different mental models of how the input data realizes an output chart. Table 2 demonstrates how users would use Data Formulator and Falx, two tools designed to alleviate the burden of data transformation in chart authoring, to create a dumbbell chart. They both operate under the same authoring paradigms: instantiate (picking a chart type in Data Formulator and choosing an inferred chart design in Falx) and delegate (Data Formulator and Falx send queries to LLMs and program synthesis algorithms to transform data, respectively). Authors experience a notable difference in the abstraction between these two tools. With Falx, they express the desired chart using a miniature version of the target chart - through visualization-by-demonstration they abstractly construct the chart. In Data Formulator, instead of constructing, authors pick a chart type and then fills in the shelf configuration with existing or desired data columns (called *data concepts*). When necessary, authors refine descriptions for data concepts using natural language or table editing, and the system generates data transformations to realize the desired data concepts. Since Falx does not expose data concepts as an abstraction, authors have no access to the inferred data transformation logic. In contrast, Data Formulator presents the generated data transformation code along with the transformed table, supporting authors to identify and correct potential mistakes from the delegated data transformation task. Interpreting data transformation code is a skill needed when using Data Formulator, but not Falx.

Authoring the same chart using the same tool. Even when we use the same tool to author the same chart, it is possible to have different authoring processes that involve different skills. Consider Line Chart Example to create a faceted line chart using ggplot2 (Table 3). The faceted line chart shows stock prices (y-axis) during the same time periods (x-axis) for three companies: IBM, Microsoft (MSFT), and Amazon (AMZN). ggplot2 incorporates methods to manipulate input data and output charts, supporting alternative design approaches to authoring the same chart with two different input data schemas. Given data in the long format with "date", "company", and "price" columns, authors can employ the `facet_grid` operator to create faceted sub line charts with the column "company"; if the input data is in the wide format with "date", "MSFT", "AMZN", and "IBM" columns, authors can apply different configurations to the same base chart using `lappy` to produce a repeated line chart for each company.

In Ribbon Chart Example, Charticulator provides two ways to create a ribbon chart (Table 3): by constructing from scratch (*paradigm*: compose), or by using a ribbon chart template (*paradigm*: instantiate). In this example, a ribbon chart shows mobile operating system market share from 2009 to 2016. The input data includes "operating system", "year", and "share" columns. For an author to compose this ribbon chart, they start by encoding a rectangle glyph's height by "share" and the x-axis of the chart by "year". Then they encode the "operating sytem" to the color and apply a Stack Y to the glyphs and order the stack by "share." Finally to create a ribbon chart, the author applies a "Band" link, connected by the "platform" data attribute. The input data needs to be tool-compatible with Charticulator to compose the ribbon chart from scratch. However, in the other use case – to instantiate the chart from a template – the input data needs to be exactly compatible with the given template. Consequently, the abstractions and expressions involved are different for creating the same chart with the same tool.

Different abstractions using the same tool. Finally, we consider a scenario which further demonstrates how different dimensions in the CAPED framework can be combined in unconventional chart authoring approaches. First, in StructGraphics [43], the authoring process consists of two stages (Table 4): 1) users start by composing graphics on a canvas without the prerequisite of an input dataset, the graphical attributes of the composed graphics form a data table structure that can be assigned to data, like a template; 2) the author then instantiates this bespoke template they authored in Structgraphics by applying compatible data to it. In the first stage, the chart abstraction is data-bound vector-graphics, accomplished through a drawing interface; in the second stage, the abstraction is a template.

Similar to StructGraphics, Mystique [12] derives a data table structured from vector graphics, sharing the instantiation paradigm and template-compatible input data with the StructGraphics approach (Table 4). However, Mystique does not require users to have the skills to draw desired charts in a canvas. Instead, it uses a mixed-initiative approach to analyze the graphical structure in an existing SVG chart. As a result, users need to be able to specify the mappings between data attributes and visual properties through a wizard interface composed of interactive widgets.

Output Chart	Tool	Input Data	Paradigm	Abstraction	Expression
faceted line chart	ggplot2	long-format data	compose	building block construction	declarative DSL
		wide-format data	compose	building block construction	declarative + imperative DSL
ribbon chart	Charticulator	tool-compatible data	compose	building block construction	drawing, widget manipulation
		template-compatible data	instantiate	parameterized template	widget manipulation

Table 3: Two examples of authoring the same chart (faceted line chart and ribbon chart) using the same tool (ggplot2 and Charticulator) but following different authoring approaches due to the differences in the input data.

Output Chart	Tool	Input Data	Paradigm	Abstraction	Expression
Sankey diagram	StructGraphics	not required → template-compatible data	compose → instantiate	data-bound graphics → parameterized template	drawing → table editing
tower chart	Mystique	template-compatible data	instantiate	parameterized template	widget manipulation

Table 4: Additional examples of unconventional chart authoring approaches given unique output charts (Sankey diagram and tower chart).

6.1.2 Describing Skills as Knowledge

One way to capture the skills required for chart authoring is to identify the conceptual, procedural, and meta-cognitive knowledge associated with the tasks. We acknowledge that it is impossible to enumerate all possible knowledge for all the lower-level tasks because of the "infinite regress problem" [22]. However, it is still beneficial to provide concrete examples of these different types of knowledge to ground our discussion. Table 5 includes some example knowledge for each of the four authoring paradigms, focusing on the tools. We formulate knowledge in the form of verb + noun phrases, following the convention used in education and learning science [7, 21].

Conceptual knowledge involves understanding the meaning of chart elements and the frameworks that underlie chart creation. The ability to understand the kind of data required by a chosen tool (e.g., tidy, tool-compatible), and the ability to accurately interpret the chart to be created are essential for all four authoring paradigms. To be able to *compose* a chart using a chosen tool, users must understand the meaning of the primitives provided by the tool and the relationships between them. They should also be able to give examples for each primitive. To *instantiate* a chart, users must be able to know whether the desired template to create the chart is provided by the tool, and understand what a desired template means. To *delegate* the creation task, users should be able to reliably estimate the capability of the delegate and the quality of the output, as well as to correctly interpret the results generated by the tool. Finally, to *repurpose* an existing chart, users must understand the underlying mechanisms in the tool for the creation of both the source chart and the target chart.

Procedural knowledge refers to the ability to carry out the concrete steps needed to author charts using a tool's interface. A common type of procedural knowledge shared across the authoring paradigms is the ability to prepare data in the required input format. In the *compose* paradigm, this involves executing the construction steps by combining the primitives through the tool interface. With *instantiate*, procedural knowledge is required to identify and apply templates, and then populate them with data through the user interface. In the *delegate* paradigm, an example of procedural knowledge is to specify the desired chart using the language (e.g., natural language, demonstration) accepted by the tool. For *repurpose*, users need to be able to re-specify the target design or make adjustments through the interface.

Meta-cognitive knowledge encompasses awareness of the strengths and limitations of different authoring options, and the ability to make strategic choices for a specific authoring goal. For example, users need to know the capabilities and limitations of the tool and choose an appropriate tool. Once a tool is chosen, users should understand the pros and cons of different construction strategies and choose an appropriate one. When the authoring process stalls, meta-cognitive knowledge entails making a decision to switch to a different tool for a better outcome or adapt to different strategies for error recovery.

6.2 Identifying New Research Opportunities

To minimize the skills and time required for chart authoring, many researchers and technologists envision an authoring process, where users simply describe their goal (sometimes with an example image) and AI generates the desired chart in a single step. Such a process focuses on a specific combination of three dimensions in the CAPED framework: delegation (*paradigm*) + monolithic prototype (*abstraction*) + natural language (*expression*). However, due to practical constraints (e.g., incomplete or ambiguous user prompts, limited model performance, high task complexity), AI may fail to generate the correct chart automatically. More broadly, the rapid emergence of diverse chart authoring tools, including AI-powered systems, raises new questions about how different users should be supported, both through intelligent authoring systems and through visualization education. The CAPED framework provides a conceptual tool for reasoning about different aspects of authoring skills and identifying opportunities for supporting users with varying backgrounds and abilities. Below, we discuss three examples.

Opportunity 1. For users who can interpret AI-generated results, having skills related to compose (*paradigm*) and building block construction or data-bound vector-graphics (*abstraction*), we argue that it is important to support seamless transitions between different *paradigms*, *abstractions*, and *expressions*. Natural language can be an effective type of *expression* to ask AI to generate initial chart specification or code. However, it is not always the most appropriate *expression* for effectively communicating intent (e.g., describing bespoke *outputs*), or for tasks such as refining mark properties (e.g., increasing bar width) and chart styles (e.g., changing color schemes). Other forms of *expressions*, such as drawing and widget manipulation, may be more effective in certain situations. To transition from natural language to these expressions, the corresponding *paradigms* and *abstractions* may need to change as well. For example, to support easy refinement of mark properties, AI-generated D3 code (where the underlying *abstraction* is data-driven graphics) needs to be adapted into building block construction, so that users can directly manipulate the properties through a dynamically generated GUI.

Opportunity 2. The CAPED framework offers an opportunity in chart pedagogy by helping educators tailor learning materials to different tools and learners' prior skills. Rather than treating chart authoring as a fixed sequence of steps, the framework can distinguish the skills that learners need to acquire from those already supported by a tool. For example, a business professional moving from Excel to Power BI may already be familiar with fitting data into parameterized templates (*abstraction*) and manipulating widgets (*expression*). For this type of learner, instructional materials can focus less on chart types or data transformation, but more on the compose *paradigm* and visual encoding concepts needed to create more expressive charts. The same applies for introducing users to new tools such as Mystique [12] and Data Formulator [46, 47]. In these cases, learning materials need not

	Compose	Instantiate	Delegate	Repurpose
Conceptual Knowledge (examples)	For the data: Understand the data requirements of the tool and the visualization to be created.			
	Understand the meaning, application, and relationship of primitives.	Understand the meaning of the desired template and its availability.	Assess the delegate tool's capability, results, and output quality.	Understand the underlying frameworks for the source and the target.
Procedural Knowledge (examples)	For the data: Prepare the data in the required input format.			
	Conduct the required steps to construct a chart with primitives.	Identify and instantiate the appropriate template.	Specify the chart with valid semantics and syntax for the given language.	Convert the source chart using the target data.
Meta-cognitive Knowledge (examples)	Know the capabilities and limitations of the tool and choose an appropriate tool. Understand the strengths and weaknesses of different authoring strategies and choose an appropriate one. Decide to switch to a different tool for a better outcome or adapt to different strategies for error recovery.			

Table 5: Example knowledge for each of the four authoring paradigms, focusing on the tools and including considerations for the data.

emphasize manual data transformation or chart specification when these tasks are automated, but instead should help users develop the remaining skills required to effectively guide, interpret, and refine chart designs. More broadly, CAPED provides educators with a vocabulary for reasoning about learning objectives across authoring tools, enabling curricula to emphasize enduring authoring competencies rather than tool-specific procedures.

Opportunity 3. In the third example, consider a graphic designer who does not know how to code—building block construction (*abstraction*) and Domain-Specific Language (*expression*). Like the business professional in the second example, they are familiar with parameterized template (*abstraction*) and widget manipulation (*expression*) but lack skills to perform the necessary data transformations to prepare their data into the expected input data format (*input data*). However, they are likely to be able to describe the target chart (*output chart*) in natural language instruction (*expression*) and more skilled in conveying the target chart with free-form drawing (e.g., sketching using a pen, manipulating shapes using a mouse). In addition, they are comfortable working at the level of vector graphic primitives like shapes and paths (*abstraction*). Existing tools like DataInk [52] and StructGraphics [43] introduce interaction techniques to support users with these skills, but they do not help with the necessary tasks (i.e., data transformation) they lack skills in. Unlike the second example where AI can generate a parameterized template from drawing, in this case, the designers may benefit more from a tighter integration between AI and drawing interfaces that allow them to create bespoke charts.

7 Discussion and Future Work

CAPED is the first theoretical framework to characterize user skills in static chart authoring through the lens of tasks and tools. In this section, we reflect on the framework's characteristics, scope, and limitations, as well as the opportunities for future research.

The five dimensions in our framework are not strictly orthogonal: categories across dimensions may exhibit interdependencies and correlations. For example, natural language instruction as an *expression* is often associated with the delegate *paradigm* and the monolithic prototype *abstraction*. However, as discussed in Section 5, associations between specific dimensions' categories are common but not deterministic, and no dimension can be reduced to another. We emphasize that such relationships between the dimensions do not constitute a limitation. Many influential frameworks in HCI and visualization similarly employ dimensions that are conceptually distinct yet partially overlapping. For example, in the Cognitive Dimensions of Notations framework [6, 18], we can identify dimensions that interact with one another (e.g., abstraction and visibility, consistency and viscosity). The observed interdependencies reflect the intrinsic coupling among the tool and task considerations in chart authoring.

We note that the CAPED framework is neither exhaustive nor definitive. As we mentioned in Section 2.1, we chose not to consider interaction and animation to have a manageable scope in this first attempt to characterize authoring skills. Authoring data visualization with interaction support would require different skills depending on the complexity of interaction and its underlying visualization. For example, some modern visualization libraries (e.g., Highcharts, Plotly) provide tooltips by default, and thus do not require additional skills to include tooltips through a simple mouseover interaction. Popular business intelligent systems with visualization support (e.g., Microsoft Power BI, Tableau) offer a simple way to include basic interactivity, such as filtering, drill-down, and brushing & linking. Building highly interactive systems to support sophisticated data exploration would require programming and beyond, such as user interface and interaction design. Future work could extend the CAPED framework to encompass animation and interaction. Doing so would likely require revising the *output chart* dimension or introducing additional dimensions to capture non-static charts, as well as expandinging the *expression* dimension to include programming languages as another form of expression.

Furthermore, we acknowledge that the current framework is primarily derived from the analysis of chart authoring tools within the information visualization domain. As a result, the five identified dimensions may not fully capture the abstractions employed in scientific or spatial visualization systems, such as the Visualization Toolkit (VTK) [38] and Paraview [2]. Data-flow pipelines, computational operators, and spatial data transformations important to these systems substantially differ from the chart-centric authoring systems analyzed in this work. Therefore, extending and validating this framework to encompass scientific and spatial visualization remains an open area for future exploration.

The framework identifies a set of core dimensions for chart authoring skills, and can serve as a foundation for future work to investigate how to measure or quantify skill levels along one or more of these dimensions. For example, researchers could investigate how to assess prompting proficiency in chart authoring systems, or to systematically evaluate candidates' expertise with a visualization library. In addition, researchers could develop standardized assessments similar to literacy or proficiency tests. Such assessments would have several practical applications. They could enable more principled recruitment of participants for user studies, ensuring clearer grouping of participants. They could also provide clearer criteria for job descriptions, as well as for candidate screening and skill evaluation. Overall, CAPED opens new opportunities for systematically understanding, assessing, and operationalizing chart authoring skills.

Acknowledgments

Bongshin Lee was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korean Government (MSIT), Artificial Intelligence Graduate School Program, Yonsei University, under Grant RS-2020-II201361. Zhicheng Liu was supported in part by NSF grant IIS-2239130.

References

[1] Tableau: Business Intelligence and Analytics Software. 5, 6

[2] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717(8), 2005. doi: 10.1109/38.865875 9

[3] R. Amar, J. Eagan, and J. Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization*, pp. 111–117, 2005. doi: 10.1109/INFVIS.2005.1532136 3

[4] P. Attewell. What is skill? *Work and occupations*, 17(4):422–448, 1990. doi: 10.1177/0730888490017004003 4

[5] S. Beschi, D. Falessi, S. Golia, and A. Locoro. Characterizing data visualization literacy for standardization: A systematic literature review. *IEEE Access*, 13:65704–65725, 2025. doi: 10.1109/ACCESS.2025.3559298 1, 3

[6] A. F. Blackwell, C. Britton, A. Cox, T. R. Green, C. Gurr, G. Kadoda, M. S. Kutar, M. Loomes, C. L. Nehaniv, M. Petre, et al. Cognitive dimensions of notations: Design tools for cognitive technology. In *International Conference on Cognitive Technology*, pp. 325–341. Springer, 2001. doi: 10.1007/3-540-44617-6_31 9

[7] B. S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, D. R. Krathwohl, et al. *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain*. Longman New York, 1956. 4, 8

[8] J. Boy, R. A. Rensink, E. Bertini, and J.-D. Fekete. A principled way of assessing visualization literacy. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1963–1972, 2014. doi: 10.1109/TVCG.2014.2346984 3

[9] M. Brehmer and T. Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013. doi: 10.1109/TVCG.2013.124 3, 4

[10] A. Burns, C. Lee, R. Chawla, E. Peck, and N. Mahyar. Who do we mean when we talk about visualization novices? In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–16, 2023. doi: 10.1145/3544548.3581524 1, 3

[11] S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, Feb. 1999. 4

[12] C. Chen, B. Lee, Y. Wang, Y. Chang, and Z. Liu. Mystique: Deconstructing SVG Charts for Layout Reuse. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):447–457, 2023. doi: 10.1109/TVCG.2023.3327354 5, 6, 7, 8

[13] W. Cui, J. Wang, H. Huang, Y. Wang, C.-Y. Lin, H. Zhang, and D. Zhang. A Mixed-Initiative Approach to Reusing Infographic Charts. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):173–183, 2022. doi: 10.1109/TVCG.2021.3114856 5

[14] M. Elias and A. Bezerianos. Exploration views: understanding dashboard creation and customization for visualization novices. In *IFIP Conference on Human-Computer Interaction*, pp. 274–291. Springer, 2011. doi: 10.1007/978-3-642-23768-3_23 3

[15] L. W. Ge, Y. Cui, and M. Kay. Avec: An assessment of visual encoding ability in visualization construction. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, article no. 1166, 16 pages, 2025. doi: 10.1145/3706598.3713364 3

[16] L. Grammel, M. Tory, and M.-A. Storey. How information visualization novices construct visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):943–952, 2010. doi: 10.1109/TVCG.2010.164 3

[17] F. Green. *What is Skill? An Inter-Disciplinary Synthesis*. Centre for Learning and Life Chances in Knowledge Economies and Societies London, 2011. 4

[18] T. R. Green. Cognitive dimensions of notations. *People and computers V*, pp. 443–460, 1989. 9

[19] J. Hong, C. Seto, A. Fan, and R. Maciejewski. Do llms have visualization literacy? an evaluation on modified visualizations to test generalization in data interpretation. *IEEE Transactions on Visualization and Computer Graphics*, 31(10):7004–7018, 2025. doi: 10.1109/TVCG.2025.3536358 1, 3

[20] S. Huron, S. Carpendale, A. Thudt, A. Tang, and M. Mauerer. Constructive visualization. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, p. 433–442, 2014. doi: 10.1145/2598510.2598566 3

[21] D. R. Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002. doi: 10.1207/s15430421tip4104_2 4, 8

[22] K. Kusterer. Workplace knowhow: The important working knowledge of unskilled workers. *Boulder, CO: Westview*, 1978. 4, 8

[23] F. D. Le Deist and J. Winterton. What is competence? *Human resource development international*, 8(1):27–46, 2005. doi: 10.1080/1367886042000338227 4

[24] S. Lee, S.-H. Kim, and B. C. Kwon. Vlat: Development of a visualization literacy assessment test. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):551–560, 2017. doi: 10.1109/TVCG.2016.2598920 1, 3

[25] Z. Liu, C. Chen, and J. Hooker. Manipulable Semantic Components: A Computational Representation of Data Visualization Scenes. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):732 – 742, 2025. doi: 10.1109/TVCG.2024.3456296 6

[26] Z. Liu, J. Thompson, A. Wilson, M. Dontcheva, J. Delorey, S. Grigg, B. Kerr, and J. Stasko. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2018. doi: 10.1145/3173574.3173697 6

[27] G. Lupi, S. Posavec, and M. Popova. *Dear Data: A Friendship in 52 Weeks of Postcards*. Princeton Architectural Press, New York, illustrated edition ed., Sept. 2016. 2

[28] J. Mackinlay, P. Hanrahan, and C. Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, 2007. doi: 10.1109/TVCG.2007.70594 6

[29] A. M. McNutt and R. Chugh. Integrated Visualization Editing via Parameterized Declarative Templates. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, number 17, pp. 1–14. 2021. doi: 10.1145/3411764.3445356 6

[30] G. G. Méndez, U. Hinrichs, and M. A. Nacenta. Bottom-up vs. top-down: Trade-offs in efficiency, understanding, freedom and creativity with infovis tools. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, p. 841–852, 2017. doi: 10.1145/3025453.3025942 3

[31] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009. doi: 10.1109/TVCG.2009.111 3

[32] T. Munzner. *Visualization Analysis and Design*. CRC Press, 2014. 3

[33] D. Ren, B. Lee, and M. Brehmer. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):789–799, 2018. doi: 10.1109/TVCG.2018.2865158 4, 6

[34] D. Ren, B. Lee, M. Brehmer, and N. H. Riche. Reflecting on the evaluation of visualization authoring systems : Position paper. In *2018 IEEE Evaluation and Beyond - Methodological Approaches for Visualization (BELIV 2018)*, pp. 86–92, 2018. doi: 10.1109/BELIV.2018.8634297 3

[35] A. Satyanarayan and J. Heer. Lyra: An interactive visualization design environment. In *Computer Graphics Forum*, vol. 33, pp. 351–360. Wiley Online Library, 2014. doi: 10.1111/cgf.12391 6

[36] A. Satyanarayan, B. Lee, D. Ren, J. Heer, J. Stasko, J. Thompson, M. Brehmer, and Z. Liu. Critical reflections on visualization authoring systems. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):461–471, 2019. doi: 10.1109/TVCG.2019.2934281 3, 5

[37] 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, 2016. doi: 10.1109/TVCG.2016.2599030 6

[38] W. J. Schroeder, L. S. Avila, and W. Hoffman. Visualizing with VTK: a tutorial. *IEEE Computer graphics and applications*, 20(5):20–27, 2000. 9

[39] H. Shi, Y. Wang, J. Chen, C. Wang, and B. Lee. PiCCL: Data-Driven Composition of Bespoke Pictorial Charts. *IEEE Transactions on Visualization & Computer Graphics*, 2025. doi: 10.1109/TVCG.2025.3634264 6

[40] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, pp. 336–343, 1996. doi: 10.1109/VL.1996.545307 3

[41] M. Solen, S. Pandey, F. Moosvi, A. Ottley, and T. Munzner. Visualization literacy or skillset? beyond the analogy to textual literacy. In *2025 IEEE Seventh Workshop on Visualization for Communication (VisComm)*, pp. 22–27, 2025. doi: 10.1109/VisComm69388.2025.00009 3

[42] M. Tory and T. Moller. Rethinking visualization: A high-level taxonomy. In *IEEE Symposium on Information Visualization*, pp. 151–158, 2004. doi: 10.1109/INFVIS.2004.59 3

[43] T. Tsandilas. StructGraphics: Flexible Visualization Design through Data-Agnostic and Reusable Graphical Structures. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/TVCG.2020.3030476 6, 7, 9

[44] E. R. A. Valiati, M. S. Pimenta, and C. M. D. S. Freitas. A taxonomy of tasks for guiding the evaluation of multidimensional visualizations. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, p. 1–6, 2006. doi: 10.1145/1168149.1168169 3

[45] C. Wang, Y. Feng, R. Bodik, I. Dillig, A. Cheung, and A. J. Ko. Falx: Synthesis-Powered Visualization Authoring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2021. doi: 10.1145/3411764.3445249 5, 6, 7

[46] C. Wang, B. Lee, S. M. Drucker, D. Marshall, and J. Gao. Data Formulator 2: Iterative creation of data visualizations, with ai transforming data along the way. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2025. doi: 10.1145/3706598.3713296 8

[47] C. Wang, J. Thompson, and B. Lee. Data Formulator: AI-powered Concept-driven Visualization Authoring. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1128–1138, 2024. doi: 10.1109/TVCG.2023.3326585 5, 7, 8

[48] Y. Wang, Z. Hou, L. Shen, T. Wu, J. Wang, H. Huang, H. Zhang, and D. Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2023. doi: 10.1109/TVCG.2022.3209357 3

[49] H. Wickham. *ggplot2: Elegant graphics for data analysis*. Springer, 2009. 6

[50] H. Wickham. Tidy data. *Journal of statistical software*, 59(10):1–23, 2014. doi: 10.18637/jss.v059.i10 5

[51] L. Wilkinson. *The Grammar of Graphics*. Springer, New York, 2nd edition ed., July 2005. 6

[52] H. Xia, N. Henry Riche, F. Chevalier, B. De Araujo, and D. Wigdor. Dataink: Direct and creative data-oriented drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2018. doi: 10.1145/3173574.3173797 6, 9

[53] J. S. Yi, Y. a. Kang, J. T. Stasko, and J. A. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007. doi: 10.1109/TVCG.2007.70515 4