VizCoach: Designing an Orchestration-Based Tool for Data Visualization Education

Shubham Chawla
Årizona State University
schawl17@asu.edu

Michael Kintscher
Årizona State University
mkintsch@asu.edu

Jai Narula
Årizona State University
jnarula1@asu.edu

Anjana Arunkumar
Northeastern University
a.arunkumar@northeastern.edu

Ashish Amresh
Northern Arizona University
ashish.amresh@nau.edu

Chris Bryan
Årizona State University
cbryan16@asu.edu

Abstract

Visualization courses are commonly found in university settings, but many students (particularly those from computer science or STEM backgrounds) struggle learning the "design thinking" aspects of visualization. In this paper, we investigate how to design and engineer a technology-enhanced learning platform for teaching visualization concepts called VizCoach. VizCoach adopts a hands-on, orchestration-based approach that abstracts away the coding aspects of visualization construction, allowing students to focus on applying design thinking principles during learning. VizCoach also supports instructors by providing workflows tailored for creating and moderating learning activities. Empirical evaluations help validate that VizCoach supports design and engineering requirements for successful orchestration and design thinking learning scenarios. We also discuss how tools like VizCoach contribute to technology-enhanced learning for visualization, and can provide opportunities for future research into visualization learning processes.

Keywords: visualization, design thinking, orchestration, technology-enhanced learning

1. Introduction

Today, university computer science (CS) and STEM departments regularly teach on data visualization, either as a standalone course or integrated into a larger data science or analytics curriculum (Liu et al., 2023). As a discipline, visualization integrates knowledge and methods from a wide range of domains including CS, statistics, design, psychology, cognitive science, and more (Bach et al., 2024). Within CS/STEM-based visualization courses, students generally come from a math or engineering background, and are generally prepared to learn the programming aspects of visualization (e.g., writing code to render visualizations on a computer), but they tend to have little (if any) background in design theory or artistic principles (Meyer & Norman, 2020). This need is well recognized in the visualization education community, as prior work has shown that CS/STEM students can struggle with the creative mindset required to reason about and apply visualization design theories and principles, especially when going beyond simple charts (e.g., basic scatter plots or bar charts) (Kerren, 2013).

One strategy for addressing this is technology-enhanced learning (TEL) (Kirkwood & Price, 2014), such as using software tools to scaffold teaching and learning. For the current paper, we are interested in investigating how to design and engineer a TEL tool that can emphasize the "design thinking" aspects of visualization via hands-on learning activities. (We define design thinking as applying and synthesizing principles and theories of visualization design when constructing and evaluating charts; such principles and theories are commonly taught in visualization curriculum and textbooks, such as choosing appropriate marks and channels for a given data and task scenario, applying the rules of expressiveness and effectiveness, etc.) Our tool, named VizCoach, adopts a web-based, orchestration approach to support real-time administration of students performing learning activities.

In particular, VizCoach develops a complementary set of workflows and views: (i) Instructors can set up classrooms, create learning activities, deploy, monitor, and review these activities in real time, communicate with students during activities, and also review and provide feedback about student work. (ii) Students

can access a corresponding set of student views, which allow them to work on and submit deployed activities, and also communicate with and review feedback from the instructor. A key feature in VizCoach's approach is that it abstracts the programming aspects of visualization construction (i.e., students are not required to perform low-level coding). Instead, the tool supports constructing charts in a no-code manner.

To our knowledge, such a no-code, orchestration-based approach has not previously been considered within visualization education. However, the design an engineering of such a tool is non-trivial. To scaffold this process, we employ a design study methodology (Sedlmair et al., 2012) based around distilling a set of design requirements which are realized by implementing and evaluating VizCoach. This allows us to focus on investigating questions like, "*how do we support both students and instructors by designing and engineering a TEL software for visualization design thinking education, within an orchestration context?*"

As a software artifact, VizCoach is suitable for deploying and conducting learning activities in classrooms. We distill a number of insights and lessons learned about how to successfully enable an accessible, orchestration-based approach for teaching and learning about visualization, such as what types of UI/UX workflows can be emphasized for both students and instructors, and discuss how tools like VizCoach have potential applications in other learning contexts outside of CS/STEM curriculums (e.g., business or graphic design courses). The VizCoach codebase (including installation and usage instructions) is published open source at: https://github.com/svl-at-asu/viz-coach.

2. Related Work

Visualization Education. While "information visualization" has been a recognized research discipline for several decades, visualization education itself has largely lagged (Liu et al., 2023). For example, it wasn't until the last decade that visualization textbooks (e.g., Munzner, 2014) became popular for formal classroom instruction. Further, while there are many individual examples of tools and curriculums that have been developed for teaching visualization (e.g., Bhargava et al., 2021; Diehl et al., 2021), there is less established pedagogy about how to best introduce the design principles and concepts that are applied when crafting visualizations, especially for CS/STEM students who do not have a background in design (Santos & Perer, 2020). Visualization is also used as a teaching technique in other disciplines (e.g., visualizing population demographics in a history class); though not the focus of this paper, we discuss such potential applications for VizCoach in Section 6.

Notably, recent work by Bach et al., 2024 has identified a number of open challenges and opportunities in visualization education, and enacts a "call to action" that includes developing novel TEL tools like VizCoach. Specifically, VizCoach addresses several challenges identified in that paper, including supporting the effectively dissemination and assessment of student work at scale and distance in large classrooms during learning activities (i.e., via VizCoach's orchestration and web-based approach), fostering core skills around visual representation (via a no-code approach that emphasizes design thinking), and providing an environment for hands-on, creative work. This is a primary motivation in developing VizCoach: not just to create a novel software artifact, but to proactively address open challenges faced by the visualization education community.

Classroom Orchestration. Orchestration is "*the design, adaption, and planning of a learning activity for a specific class, and the real-time management of that activity across multiple planes and under multiple constraints*" (Dillenbourg et al., 2018). Within this context, technology can enhance the planning, designing, and managing of learning activities (Dillenbourg et al., 2018). One motivation for the use of orchestration is scalability and flexibility: software-based orchestration can provide benefits to sensing and interacting with students in large (and online) classrooms (Roschelle et al., 2013). Another benefit is the potential for active and inquiry-based learning, which can lead to deeper and more sustained knowledge retention and enhance creative thinking and student engagement (Pahl & Kenny, 2008; Serrano et al., 2019); this can be especially beneficial for CS/STEM students who have less exposure to creative thinking and design-based coursework. In the context of this paper, the challenge is to understand *how to create such a TEL tool* in a way that supports visualization learning activities. Section 3 distills a set of design requirements for this, which are then operationalized in VizCoach's implementation.

3. Design Requirements Analysis

As mentioned above, the motivation for VizCoach included community calls for new tools that can emphasize the design thinking aspects of visualization (particularly for CS/STEM students) and the potential of TEL and orchestration-based approaches to support learning. To help guide VizCoach's design, we first conducted a meta-analysis of open challenges and needs in TEL for visualization education (e.g., using

Approach	Example Artifacts	Open Source	Flexible Chart Construction	Requires Programming	Supports Orchestration	Support for Authoring Learning Activities (e.g., adding instructions)	Instructions Embeddable within Activities	Real-Time Instructor Student Communication	Student Work is Stored
Data Analysis Platforms	Tableau, Power BI, Excel, CODAP, ParaView, Gephi	Yes	Yes	No	No	No	No	No	Locally
Lower-Level Vis Programming Libraries	Matplotlib, D3.js, three.js, p5.js	Yes	Yes	Yes	No	No	No	No	Locally
Higher-Level Declarative Vis Programming Libraries	ggplot2, Plotly, Seaborn, Shiny	Yes	Yes	Yes	No	No	No	No	Locally
Declarative Vis Grammars	Vega, Vega-Lite, Altair	Yes	Yes	No	No	No	No	No	Locally
Data Notebooks	Observable, R Notebooks, Project Jupyter Notebooks (incl. Google Colab)	Yes (except Colab)	Yes	Yes	No	Yes	Yes	No	Locally or online (not shared with instructor by default)
VizCoach		Yes	Yes	No	Yes	Yes	Yes	Yes	To instructor's DB

Figure 1. This matrix reviews several existing approaches to visualization creation (including example artifacts for each approach type) to understand their limitations in the context of VizCoach's design goals.

literature discussed in Section 2, including Bach et al., 2024) and reviewed popular visualization approaches and artifacts to assess their feasibility for a no-code, hands-on, orchestration-based approach to visualization education (see Figure 1). Based on this, we distilled a set of six high-level design requirements: three focused on supporting students doing learning activities (SR1–SR3) and three supporting instructors preparing and orchestrating the activities (IR1–IR3). (Please see Section 6 for a reflection on the successes of implementing these design requirements in VizCoach.)

SR1: Emphasizing the Design Thinking Aspects of Data Visualization. Design thinking is a critical component of visualization construction (Munzner, 2014), such as knowing what marks and channels to select, how and when to adhere to (or violate) established design rules, how to address complexity to promote interpretation, and more. When students are learning about how to apply rules and principles of design, forcing them to perform low-level coding and debugging could easily become a significant distractor that impedes learning. As such, VizCoach adopts a no-code approach, abstracting away the programming aspects of constructing visualizations.

SR2: Promoting Experiential and Inquiry-Based Learning. There is a large body of research that discusses how hands-on TEL tools can promote learning processes such as experiential and inquiry-based learning, with positive student outcomes including increased success, motivation, and attendance (Serrano et al., 2019). To make such processes realizable, we need an approach that supports students creating and modifying visualizations in an interactive, hands-on manner — e.g., allowing them to investigate and reflect on how charts update as their properties are changed.

SR3: Easy coordination with instructors. A key for successful visualization learning activities is that instructors are able to be aware of the states of students, and can easily review and communicate with them (Bach et al., 2024). For large and online classrooms where it is infeasible for instructors to physically walk to and talk with many students, it is necessary that students can easily communicate with and receive feedback from instructors. As VizCoach is a web-based application, such communication can be supported via real-time chat messaging and other signals between students and instructors (such as notifying an instructor when a student is finished with their work).

IR1: Accessible, Expressive, and Customizable Activity Creation and Design. On the instructor side, a critical component to successful teaching is the ability to create learning activities that students will perform. As VizCoach adopts a hands-on approach where students can create charts as part of an activity, there needs to be complementary functionality that lets instructors create these activities, including providing desired instructions and datasets that should be visualized during an activity.

IR2: Scalable Support for Larger Classroom Environments. Classrooms can be diverse, particularly in CS/STEM where many courses enroll dozens (or even 100+) students. During a learning activity, students will focus on their own individual work, but instructors need a way to overview the current state of the class, and also to "peek in" and see what individual students are doing. We develop views in VizCoach that support theses types of "summary" and "detail" viewing of students as they

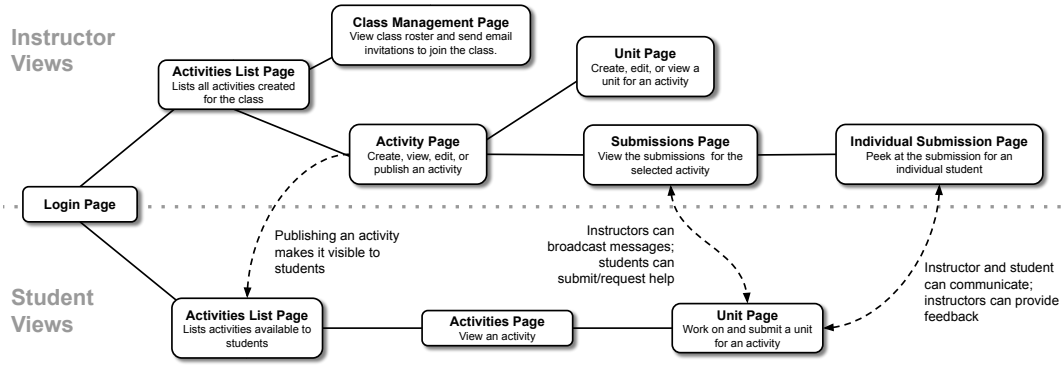


Figure 2. VizCoach supports two complementary user experiences: one for instructors who can create classrooms, design and assign activities within a classroom, and monitor/orchestrate activities. On the other side, once invited to a classroom, students can work on and submit activities and communicate with the instructor.

work on activities.

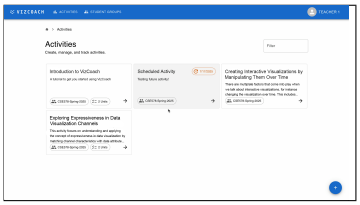
IR3: Lightweight Deployment and Administration for Instructors. Finally, while teachers who would install and use this platform are likely technically proficient (being instructors in CS/STEM courses), a software artifact that is difficult to install and maintain will ultimately result in a tool that is not used. We adopt a "lightweight" strategy for application deployment, administration, and data management, via the use of established libraries and service layers. Moreover, VizCoach is deployed as a web application and accessed via web browsers, which supports easy access and use.

4. The VizCoach System

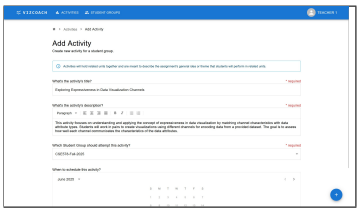
VizCoach is designed as an orchestration-based tool that supports the design requirements outlined in Section 3. The system is built around a set of complementary perspectives, summarized in Figure 2: (i) Instructors have several views for managing a classroom, including creating, editing, and deploying new learning activities, and managing/orchestrating a currently deployed activity. (ii) When students are invited to join a classroom, they log into and navigate within a set of student views that allow them to access and complete deployed activities. Below, we walk through the various pages available to each user set (system screenshots use dummy data); higher-resolution images and more detailed documentation for each page may be found in the project's GitHub repository.

4.1. Instructor Views

Login Page. Upon accessing VisCoach, a login page (not shown) prompts users to enter their email and password. If the recognized user is an instructor, the system navigates to the Activities List Page (the de facto homepage for instructors). When a student account logs in, the system navigates to the student version of the Activities List page.



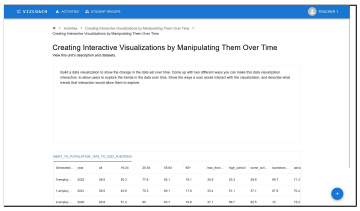
Activities List Page. On this page, an instructor can see the activities they have created for the class, including whether an activity is currently live, scheduled, or completed (closed). An instructor can select to view or edit an existing activity or create an new one. This page also links to the Class Management Page, where they can manage the roster for the class (e.g., inviting new students to join).



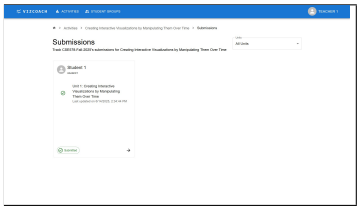
Activity Page. Activities can be viewed, edited, and created in this page. Specifically, an activity is defined as a set of instructions (e.g., a high level description of the activity and its goals) along with one or more attached units, which represent a discrete item of work that a student performs (e.g., creating a chart, reflecting on a chart via writing a response, etc.). Toggling to an 'edit' version of the Activity Page allows instructors to edit the activity's description using a rich text editor (this is shown in the inset image above), and instructors can add new units to the activity (units are edited via the Unit Page, see below). VizCoach organizes units as a list with the assumption that an activity is an ordered progression of units. For example, an activity could first ask students to consider different ways to handle complexity for a dataset, and then have them create

three different visualizations across three units, applying different techniques to handle complexity (aggregation, juxtaposition, data transformation, etc.) for each unit.

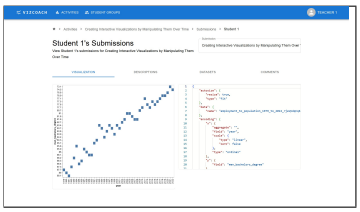
Once an activity and its associated units are ready for classroom deployment, it can be published (or scheduled to publish at a certain future time). When published, an activity and its units are visible to students logged into VizCoach, where they may access and work on them.



Unit Page. As mentioned above, units make up discrete items of work for an activity. A unit contains two elements: (i) a set of instructions and (ii) one or more attached datasets. Similar to how editing is done on the Activities Page, the Unit Page can be toggled to provide a rich text editor that allows instructors to enter instructions for the unit. Instructors can upload one or more datasets (stored as CSV files) which become available to students when they access the unit. It is not required that all units in an activity maintain the same set of datasets; each unit may have its own individual one(s). When a unit is saved, the instructions and attached datasets are shown on the Activity Page.

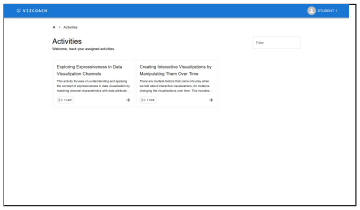


Submissions Overview Page. When an instructor publishes a learning activity and students begin working on it, the Submissions Overview Page allows instructors to review the current state of the class. Specifically, this page shows the current state of each student's work via a small multiples display. Each student's current progress is shown inside a box, allowing the instructor to quickly skim through and review the progress (or state) of each student. On the student side, students have several available actions to signal the instructor (such as "Raising a Hand" or submitting a unit); doing so adds a notification icon to that student's box and moves it to the top of the Submissions Overview Page, so the instructor can notice and take appropriate action (such as seeing why the student needs help). This page also supports re-ordering the student boxes, such as via student last name or submission time. For activities with multiple units (i.e., where students would create multiple charts), the page can be set to display only one selected unit's work, or display the unit currently being worked on by each student. If a student messages an instructor, a chat notification icon is also appended to their student box and it is shifted to the top of the page (similar to the "Raise Hand" action).

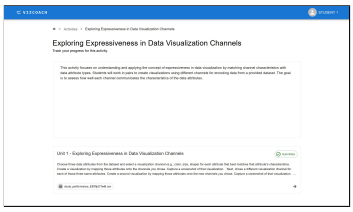


Individual Submission Page. To review in detail a student's work, an instructor can click a student's box in the Submissions Overview Page to go to the Individual Submission Page. This page shows a view similar to the student's Unit View (described below), allowing the instructor to "peek in" on the student's work (the main difference is that the chart builder functionality, which students use to construct visualizations, is disabled for instructors). This page also provides a messenger tab, where an instructor and student can send messages back and forth (e.g., to chat when a student raises their hand, or to provide feedback about a submitted unit).

4.2. Student Views



Activities List Page. When students log into VizCoach, they are taken to their version of the Activities List Page. This page lists all currently open activities for a student (the inset image shows two example activities), and allows them to select and work on an activity.



Activity Page. The student version of the Activity Page shows the instructions for the selected activity, including the summary descriptions for each of the units in the activity. Selecting a unit allows students to access and work on it. When a unit has been completed and submitted, this is indicated on the page via a "Submitted" checkmark icon.

Unit Page. The Unit page (Figure 3) is where students work on and submit a unit in an activity. This page contains four tabs: (i) the visualization tab shows the visualization that is being constructed by the student for the unit; (ii) the description tab shows the unit's instructions; (iii) the datasets tab shows a tabular spreadsheet view of the datasets for the unit; (iv) the comments tab shows messages with the instructor.

On the visualization tab (as shown in Figure 3),

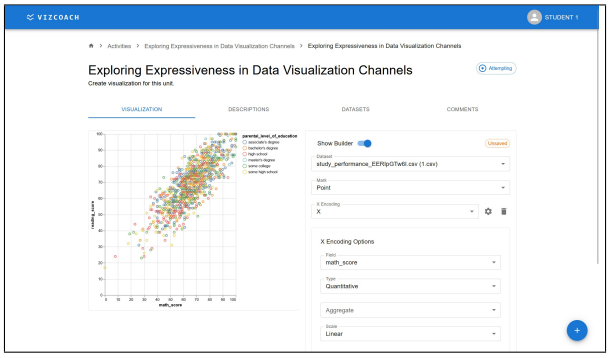


Figure 3. For students, the Unit Page includes an interactive no-code chart builder for creating Vega-Lite visualizations, along with tabs to view the unit's instructions, review its attached dataset(s) in a tabular format, and chat with the instructor.

there are two panels: the right chart builder panel provides an interactive set of dropdowns and is used to construct Vega-Lite visualizations (Satyanarayan et al., 2016). Vega-Lite is a declarative grammar-based approach for visualization construction and design, and flexibly supports defining a chart's properties and visual encodings using key-value pairs. Vega-Lite supports a broad and flexible range of visualizations, including "traditional" charts (scatter plots, bar charts, etc.) and more complex/nuanced designs. Vega-Lite also supports significant customization, including the ability to apply statistical and data transformations to the data being visualized, customizing how encodings and chart stylings are shown, and more.

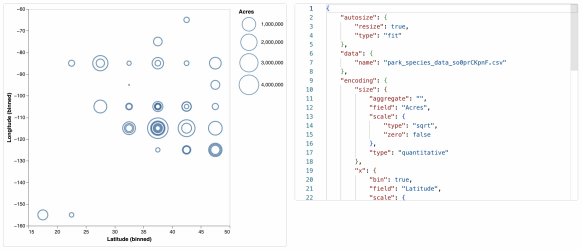


Figure 4. An example of showing the raw Vega-Lite specification in the Unit Page's chart builder, with the corresponding constructed chart on the left. This approach provides significant flexibility and customization when creating charts.

The design of this panel was inspired from an existing chart builder tool from Data.world; as options are selected, a Vega-Lite specification schema is dynamically created (likewise, as new chart options become available, appropriate dropdowns are interactively added to the chart builder panel). Once the Vega-Lite schema becomes "valid" to render a chart (i.e., sufficient properties have been defined that a chart can successfully be created), the left visualization panel begins to show the constructed chart. As additional updates are made in the chart builder panel (further updating the Vega-Lite specification), the visualization updates accordingly. Behind the scenes, Vega-Lite's renderer engine parses and processes the constructed schema when updates are made, and re-renders the chart in the left panel. This approach relieves students of having to manually code visualizations, though if desired students can toggle the chart builder panel to show the "raw" Vega-Lite specification (i.e., the actual key-value pairs describing the chart) and perform edits this way (as shown in Figure 4).

On the Unit Page, students can also trigger a "Raise Hand" notification from this page, which displays on the instructor's Submission Page to support orchestration-based awareness.

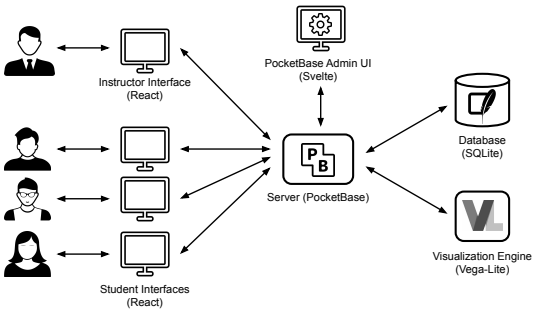


Figure 5. Overview of the platform architecture.

4.3. System Architecture Details

VizCoach is implemented as a web application suitable for deployment on either local or public hosts (e.g., Netlify, Google Cloud Platform, etc.). Figure 5 shows a high-level overview of the platform's components and how they are connected. The instructor and student frontends are implemented in React with styling primarily via Material UI. The server is a PocketBase instance written in Go. The instructor and student frontends communicate with the server using PocketBase's JavaScript SDK, and data is stored using an embedded SQLite database. An admin interface (written using the Svelte framework) allows instructors or support staff (e.g., an IT specialist setting up VizCoach for a class) to configure the database, API rules, and listen to live updates as they are made to the database. Student can set (and reset) passwords for their accounts, and only instructors are allowed to access backend modules and data, to preserve student privacy.

5. Evaluation

To evaluate VizCoach, we performed a complementary set of evaluations with the goal of demonstrating how the platform can support SR1–SR3 and IR1–IR3 outlined in Section 3. First, a student-focused evaluation assessed the usability of the platform, including validating the overall design, functionalities, affordances, encodings, and interactions for constructing and reasoning about visualizations during learning activities that were intended to promote "design thinking" aspects of visualization. For this, we recruited a set of participants representative of the skills and experience levels of students who would use VizCoach in a classroom, and had them complete two learning activities where they constructed and reflected on visualizations.

The second evaluation focused on instructor perspectives, to understand how VizCoach's no-code, hands-on, and orchestration-based approach might support their own teaching methods and classroom environments. For this, we conducted demo sessions and semi-structured interviews with three U.S.-based visualization faculty who teach data visualization courses to CS/STEM students.

5.1. Student-Focused Evaluation

For the student study, we recruited eight current CS students from Arizona State University (6 men, 2 women, no one identified as another gender; age avg $= 27.9$, SD $= 3.9$) that were representative of the skills and backgrounds of "real" target users (i.e., students enrolled in a CS-based visualization course). Put another way, the participants had prior exposure in learning about the principles and concepts that are taught in CS/STEM visualization courses (though none had previously used VizCoach), and could draw upon that knowledge when performing learning activities. This meant we did not have to conduct a pre-lecture on the topics touched on by the activities (this would allow them to focus on providing feedback about the tool).

Tasks. Participants were tasked to complete two activities based on topics taught in Munzner's *Visualization Analysis & Design* textbook (Munzner, 2014), which is a popular textbook in university-level data visualization courses both at the graduate and undergraduate level. The first activity, based on the textbook's Chapter 5 (Marks and Channels), focused on exploring the effectiveness of different encodings of data attributes to channels, while the second activity, based on Chapter 11 (Manipulate View), asked them to think about interactivity and how to present data that changes over time. Critically, these activities were designed to "go beyond" simply understanding or demonstrating visualization concepts and rules (i.e., as they are traditionally presented in textbooks and lectures), but instead required participants to apply, analyze, evaluate, and synthesize design thinking principles to create and reflect on their charts. This design intentionally mimics a potential real-world usage for VizCoach, whereby visualization topics are first introduced via "traditional" means (e.g., in a lecture), and then students apply and reinforce such concepts via hands-on learning activities.

Procedure. We created the two activities and a student login in advance. Each participant took the study one-at-a-time, to allow us better moderation and feedback collection (note that this also simulates how students in a real classroom would use the application, as they would not have access to the visualizations being constructed by other students in the class). Seven of the participants completed the study over Zoom, while one completed it in person (though we did not see any effects on the results or feedback for this distinction). In all cases, the participant had full control over interacting with and manipulating VizCoach to complete the activities.

After a brief tutorial on the VizCoach system, the participant was provided with login information for the student account and proceeded to log into VizCoach to begin the activities. Participants were given ten minutes to complete each activity (simulating a classroom-based breakout session), and were encouraged to think aloud during their work. The facilitator was available to answer questions or help if the participant became stuck or frustrated, and also used the chat functionality to provide feedback to participants during the activities. Participants submitted each activity once completed, or at the conclusion of the ten minutes.

After completing both activities, participants took a brief survey based on the short version of the user experience questionnaire (UEQ-S, see Schrepp et al., 2017), which contains eight scales about the pragmatic and hedonic experiences of using a tool. They then participated in a short semi-structured interview asking free response questions about what they liked, found difficult, and wanted improved in VizCoach.

Results. All participants reported at least a moderate level of visualization familiarity and competency (most self-reported high confidence in these), which was considered sufficient requisite knowledge to take the study. All participants were familiar with the topics covered by the activities, allowing us to focus our data collection and analysis on their experience using VizCoach.

For the UEQ-S survey, our results (summarized

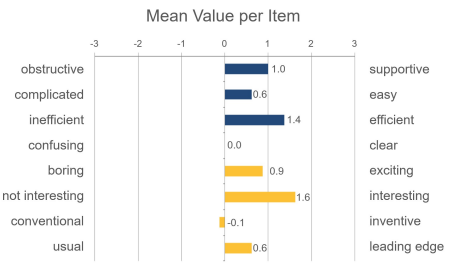


Figure 6. Summarized UEQ-S results for the student eval. Per the UEQ-S documentation, scores between -0.8 to +0.8 are considered "neutral."

in Figure 6) indicate that participants felt VizCoach's experience tended towards a "supportive," "efficient," and "interesting" workflow for performing hands-on learning activities. Interestingly, the responses indicated a rather neutral position on whether VizCoach was "conventional" or "inventive;" we consider this a positive result, as VizCoach emphasizes constructing charts in a familiar, straightforward, and efficient manner, as opposed to employing a highly novel or "leading edge" type of UI/UX workflow that the participants had to learn.

These sentiments were reinforced via comments collected via think aloud and the post-study interviews, both of which we thematically analyzed via affinity diagramming. For example, participants liked how VizCoach allowed them to seamlessly, flexibly, and interactively update the visualizations via setting chart properties in real-time, illustrated by comments like: "*This tool helps me understand the linkage between the dataset and the visualization construction process,*" and "*[i]t helped me see the connection between the data and how it appears in the chart.*"

Participant comments also included suggestions about how to improve the design of classroom activities, such as explicitly familiarizing students with datasets prior to beginning an activity: "*I should have more information about the dataset, or an opportunity to explore the dataset,*" and "*[you should] explicitly ask the student to go through the dataset first.*" Tangentially to this, participants also expressed that explicit guidance in the interface, such as tool-tips, embedded samples, and guides could be helpful while learning how to use the tool. We discuss future potential engineering enhancements for VizCoach in Section 6.

5.2. Evaluating the Instructor Views

Next, to evaluate the instructor perspective, we conducted expert feedback sessions with three U.S.-based CS professors (two assistant and one associate level), all of whom direct visualization research laboratories and teach visualization courses. We intentionally recruited each professor from a different university so we could capture broader perspectives on data visualization course design.

Procedure. Two demo activities (the same used for the student study) were set up in advance for the demo. We also demonstrated both the instructor and student capabilities in VizCoach, including creating activities and units, publishing and monitoring activities, viewing and messaging students, and working on and submitting activities (as a student). If desired, the faculty were allowed to further play and interact with VizCoach via a pair analytics approach (Arias-Hernandez et al., 2011).

After each demo, we conducted a semi-structured interview to understand their current practices for teaching visualization and solicit additional feedback about VizCoach. Refer to the supplemental materials for the full list of questions.

Results. Like the student study, affinity diagramming was used to analyze instructor feedback. All three professors valued the expressiveness of in-class activities in their current practice — e.g., two mentioned having students use pen and paper to sketch visualizations — but they found such activities difficult to scale to larger classes and difficult to facilitate in condensed course schedules. They saw significant potential in VizCoach's authoring and orchestration functionalities to support efficiently creating, moderating, and providing feedback during classroom activities at scale (e.g., in large and online courses).

Similarly, while the professors saw value in teaching coding concepts alongside visualizations (particularly since their own classrooms considered visualization programming a fundamental goal of the course), they likewise recognized that coding skills could be a barrier to teaching design thinking. One professor remarked on the challenge of separating the two when using computational notebooks (such as Project Jupyter or R notebooks) for activities: "*I try not to get the students to do five other things. That is not what the activity is [trying to teach].*" Two instructors suggested VizCoach could also approach the "visualization programming" barrier by offering a design-thinking-first approach that explicitly connects back to the underlying code: "*What if this was linked to multiple specification languages and students could see what their abstract specification looks like in different languages... From a computer science perspective I am trying to get them to see 'this is all the same stuff, the same abstraction'.*" VizCoach currently supports Vega-Lite specifications (e.g., as shown in Figure 4), though the tool could be extended to

handle additional languages or grammars in the future.

In terms of orchestration and administrative functions, the professors made several comparisons to Gradescope (e.g., such as having the ability to reuse comments when reviewing student submissions), despite it not being a visualization platform. In particular for VizCoach, the instructors liked the design of the Submission Overview Page and its support for efficiently viewing and commenting directly on student submissions (it was even suggested the feedback feature could be further expanded, such as allowing for direct annotation on top of the submitted visualizations).

6. Discussion and Conclusion

VizCoach represents a novel, no-code, hands-on, orchestration-based tool with an emphasis on designing and administering learning activities that can promote the design-thinking aspects of visualization, by employing a dual-user mode that serves both instructor and student needs. Below, we discuss some of the takeaways and lessons learned during our process of designing, implementing, and validating VizCoach, and discuss identified opportunities for future work that can build on the initial results developed in this paper.

Supporting Design Requirements SR1–SR3 and IR1–IR3. Based on our process of designing, implementing, and evaluating VizCoach, we validate that the platform supports design requirements SR1–SR3 and IR1–IR3 outlined in Section 3. SR1 is demonstrated by employing a no-code approach for chart construction, which allows students to focus on the design thinking aspects of visualization. Moreover, the ability to break down activities into separate units allows instructors to emphasize discrete aspects of design in an iterative, focused manner. Similarly, SR2 is addressed by the use of hands-on features, such as a chart builder that dynamically updates visualizations based on student interaction. SR3 is supported by the system's messaging and coordination components, such as chat functionality and the ability to signal instructors via actions such as "Raising a Hand."

On the instructor side, IR1 is supported allowing instructors to freely construct activities and units, including attaching desired datasets. IR2 is handled via VizCoach's orchestration features, including being able to see an overview of the class' current work, and being able to efficiently peek in on individual students. Finally, IR3 is supported by VizCoach's architectural design, including accessible libraries and service layers (PocketBase, Vega-Lite, etc.) and the use of web browsers to facilitate easy access.

Potential Limitations and Threats to Validity. Despite the successes of our tool in supporting Section 3's design requirements, we want to be careful about overstating our findings and acknowledge limitations and risks in the current evaluations. For example, while we contribute both a novel TEL software artifact (the VizCoach system) and several lessons learned about how to design and engineer a a no-code, hands-on, orchestration approach for visualization education, we do not directly assess learning outcomes in actual classrooms. Also, despite the overall positive feedback, our current evaluations also are composed of relatively modest sample sizes. In each student session, participants completed only two activities. These participants were likewise not currently enrolled in a visualization course (though all had previously been, and were thus representative of the skills and backgrounds of such students).

Classroom Deployments and Follow-Up Studies. We plan to conduct follow-up, longitudinal classroom deployments, and will study how VizCoach supports teaching and learning. Such studies will include robust data collection across multiple institutions and measure learning impacts and outcomes. We will also explore how VizCoach can generalize to more diverse types of learning activities, particularly when compared to traditional tools and practices.

We are also interested in understanding if VizCoach can be deployed in non-CS/STEM courses. This would include visualization courses housed in other departments (e.g., business or graphic design) and also disciplines where visualization is used to teach data-driven concepts, such as biology and mathematics.

Additional Reflections on Improving the VizCoach Platform. Finally, while our evaluations provide validations that VizCoach is feasible in its current form, we are actively maintaining and developing its codebase, including adding new features and functionalities based on feedback and suggestions from our studies. As an example of this, in the Unit Page (Figure 3) some student participants wanted the instructions on the screen at the same time as the chart builder panel, while others wanted to see the message tab instead. One strategy to achieve this is making this page more dynamic, allowing students to dynamically add or remove desired content.

Likewise, some instructors suggested providing more flexibility in how activities can be constructed and moderated (e.g., via annotated feedback, or by disabling a subset of Vega-Lite options for a specific context) as a way to support more diverse learning activities. We are interested to explore developing such features, though this needs to be done with care to ensure the overall user experience does not become overly cumbersome.

Conclusion. Visualization is widely used for understanding and communicating about data, but it is crucial that practitioners who create visualizations know how to apply design principles and theories. There is a need to develop innovative tools to scaffold such learning, but this is a non-trivial problem, particularly since solutions need to support both instructors and students. This paper investigates a novel approach for such a tool, contributing both a software artifact (the VizCoach platform) and also demonstrating how a no-code, hands-on, orchestration-based approach can be successfully implemented for visualization education. Future work will not only to continue to improve VizCoach, but we plan to study how the tool supports learning and teaching via longitudinal classroom deployments.

Acknowledgement

This research was supported in part by the U.S. National Science Foundation through grant DUE-2216452.

References

Arias-Hernandez, R., Kaastra, L., Green, T., & Fisher, B. (2011). Pair analytics: Capturing reasoning processes in collaborative visual analytics. *2011 44th Hawaii International Conference on System Sciences*, 1–10.

Bach, B., Keck, M., Rajabiyazdi, F., Losev, T., Meirelles, I., Dykes, J., Laramee, R., AlKadi, M., Stoiber, C., Huron, S., Perin, C., Morais, L., Aigner, W., Kosminsky, D., Boucher, M., Knudsen, S., Manataki, A., Aerts, J., Hinrichs, U., … Carpendale, S. (2024). Challenges and opportunities in data visualization education: A call to action. *IEEE Transactions on Visualization and Computer Graphics*, *30*(1), 649–660.

Bhargava, R., Williams, D., & D'Ignazio, C. (2021). How learners sketch data stories. *2021 IEEE Visualization Conference (VIS)*, 196–200.

Diehl, A., Firat, E., Torsney-Weir, T., Abdul-Rahman, A., Bach, B., Laramee, R., Pajarola, R., & Chen, M. (2021). VisGuided: A community-driven approach for education in visualization. *Proceedings Eurographics Education Papers*, 23–30.

Dillenbourg, P., Prieto, L., & Olsen, J. (2018). Classroom orchestration. In *International handbook of the learning sciences* (pp. 180–190). Routledge.

Kerren, A. (2013). Information visualization courses for students with a computer science background. *IEEE Computer Graphics and Applications*, *33*(2), 12–15.

Kirkwood, A., & Price, L. (2014). Technology-enhanced learning and teaching in higher education: What is 'enhanced'and how do we know? a critical literature review. *Learning, Media and Technology*, *39*(1), 6–36.

Liu, X., Alharbi, M., Chen, J., Diehl, A., Rees, D., Firat, E., Wang, Q., & Laramee, R. (2023). Visualization resources: A survey. *Information Visualization*, *22*(1), 3–30.

Meyer, M., & Norman, D. (2020). Changing design education for the 21st century. *She Ji: The Journal of Design, Economics, and Innovation*, *6*(1), 13–49.

Munzner, T. (2014). *Visualization analysis and design*. CRC Press.

Pahl, C., & Kenny, C. (2008). The future of technology enhanced active learning: A roadmap. In *Technology enhanced learning: Best practices* (pp. 348–375). IGI Global.

Roschelle, J., Dimitriadis, Y., & Hoppe, U. (2013). Classroom orchestration: Synthesis. *Computers & Education*, *69*, 523–526.

Santos, B., & Perer, A. (2020). Visualization for data scientists: How specific is it? *Eurographics (Education Papers)*, 39–43.

Satyanarayan, A., Moritz, D., Wongsuphasawat, K., & Heer, J. (2016). Vega-Lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 341–350.

Schrepp, M., Hinderks, A., & Thomaschewski, J. (2017). Design and evaluation of a short version of the user experience questionnaire (UEQ-S). *International Journal of Interactive Multimedia and Artificial Intelligence*, *4*(6), 103–109.

Sedlmair, M., Meyer, M., & Munzner, T. (2012). Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, *18*(12), 2431–2440.

Serrano, D., Dea-Ayuela, M., Gonzalez-Burgos, E., Serrano-Gil, A., & Lalatsa, A. (2019). Technology-enhanced learning in higher education: How to enhance student engagement through blended learning. *European Journal of Education*, *54*(2), 273–286.