

# Visual Analysis and Coding of Data-Rich User Behavior

Tanja Blascheck\*, Fabian Beck\*, Sebastian Baltes†, Thomas Ertl\*, and Daniel Weiskopf\*

\*University of Stuttgart, Germany

†University of Trier, Germany

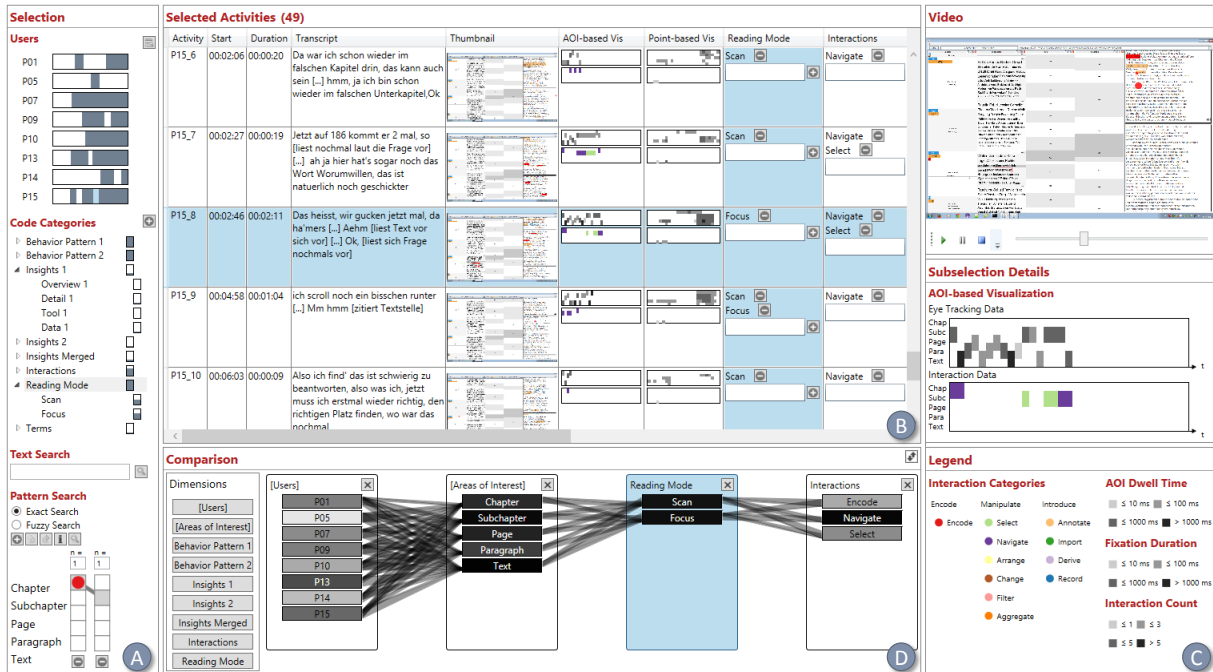


Figure 1: Interactive coding of user behavior for transcribed user studies: (A) *Selection Panel*, which lists recorded users, codes, and code categories, as well as options for searching and filtering; (B) *Selected Activities Panel*, which represents all selected user activities in a visually enriched tabular representation including transcript, word-sized visualizations of eye movement and interaction data, and assigned codes; (C) *Sidebar*, which provides additional information of a selected activity such as video, enlarged visualizations, statistics, and a legend; (D) *Comparison Panel*, which allows contrasting codes of different categories and other categorical attributes of activities.

## ABSTRACT

Investigating user behavior involves abstracting low-level events to higher-level concepts. This requires an analyst to study individual user activities, assign codes which categorize behavior, and develop a consistent classification scheme. To better support this reasoning process of an analyst, we suggest a novel visual analytics approach which integrates rich user data including transcripts, videos, eye movement data, and interaction logs. Word-sized visualizations embedded into a tabular representation provide a space-efficient and detailed overview of user activities. An analyst assigns codes, grouped into code categories, as part of an interactive process. Filtering and searching helps to select specific activities and focus an analysis. A comparison visualization summarizes results of coding and reveals relationships between codes. Editing features support efficient assignment, refinement, and aggregation of codes. We demonstrate the practical applicability and usefulness of our approach in a case study and describe expert feedback.

**Index Terms:** I.3.6 [Methodology and Techniques]: Interaction techniques—; H.5.2 [User Interfaces]: Evaluation/Methodology—

\*e-mail: firstname.lastname@vis.uni-stuttgart.de

†e-mail: research@sbaltes.com

## 1 INTRODUCTION

Analyzing the results of a user study often requires substantial manual work, in particular for qualitative data such as experiment transcripts, video, or audio recordings. The challenge for an analyst is to condense this rich but unstructured data into abstract concepts. A central element of this process is assigning codes to activities users have performed during a study. In general, coding is an essential method of formal qualitative research methodologies such as *Grounded Theory* [14, 19]. It can form the basis for a later quantitative analysis of coded user behavior as part of a *mixed-method approach* [2, 51]. In context of visualization research, for instance, North [42] brought forward the idea of identifying and coding *visualization insight* as a qualitative approach studying user behavior.

Recording user behavior within an interactive system typically includes a rich variety of data streams (e.g., video and audio recordings, transcripts of user behavior or of think-aloud protocols, eye movement data, and interaction logs). There exist software tools which support interactive coding. However, these tools only integrate some of the data, usually, transcripts and video/audio. Moreover, their abilities to support an interactive exploration and analysis of codes is limited.

In this work, we suggest a visual analytics approach for improved coding of user behavior (Figure 1) which eases the tasks processing and analyzing qualitative data. Our approach integrates various data streams, such as transcripts of user activities and video recordings or screencasts, into a tabular representation (Figure 1,

B). The table embeds eye movement and interaction data as word-sized visualizations [3], which can be enlarged for detailed exploration (Figure 1, C). We enable advanced interactive coding through the following concepts of visually supported reasoning:

- **Codes and coding categories:** An analyst assigns codes to every user activity, supported by auto-completion and collective code assignments. Codes grouped into categories build a coding scheme and reflect different levels of an analysis or results from several analysts (Figure 1, A and B).
- **Visually-enriched faceted browsing:** Our approach implements *faceted browsing* [58] to filter the table of user activities by user, codes, textual search, eye movement and interaction patterns (Figure 1, A and B). Word-sized visualizations highlight selected activities in user timelines and show covered codes and categories (Figure 1, A).
- **Code comparison visualization:** A dedicated view (Figure 1, D) facilitates an interactive comparison of coding categories and other categorical attributes of user activities. Vertical axes represent grouping dimensions; each node on an axis identifies a code or categorical value. Links between axes visualize the overlap of nodes with respect to included user activities.

Our approach is novel and goes beyond existing solutions, in particular regarding visually supported analysis, editing of codes and data integration (Section 2). Based on defined system requirements and analysis tasks, we decided to blend a mixture of algorithms, existing interaction concepts, and visualization techniques. Additionally, we compare our approach to related systems to highlight similarities and differences (Section 3). A detailed description of our system shows how we implemented our requirements (Section 4). In a case study, we use our approach to evaluate the results of a user study highlighting insights an analyst might gain when using our system (Section 5). We first collected user feedback using questionnaires, which yielded detailed informal feedback from experts (Section 6). Finally, we discuss contributions and limitations (Section 7) and draw concluding remarks (Section 8).

Please note that, throughout this paper, we consistently distinguish *users* as participants of an analyzed user study from *analysts* as people who use our approach.

## 2 RELATED WORK

*Computer-assisted qualitative data analysis software* (CAQDAS, e.g., ATLAS.ti, MAXQDA) supports coding of qualitative data collected in studies and experiments [38]. Wikipedia maintains a list of available tools<sup>1</sup> and Friese [18] compares six CAQDAS systems. While CAQDAS approaches are usually general tools to code textual data (often enriched with video recordings), some approaches are closer to our work as they specialize in analyzing user behavior and partly integrate data-rich recordings. For instance, Dou et al. [16] display interaction data in a timeline representation; annotations can be added to interaction events. SensePath [41] supports categorizing activities from web browsing and depicting them in chronological order together with a video. ChronoViz [57] is a visualization tool, which allows coding using multiple data sources (video, audio, interaction logs, sensor readings, paper notes, transcriptions, and eye movement data). In contrast to our approach, ChronoViz focuses on the video as the main data source and does not offer a visualization technique for comparing annotations constructed during a coding process.

There are visualization approaches without coding support, which represent coding behavior. If data is already coded beforehand, for instance, a timeline may show coded activities, summarizing quantities [49] or focusing on relations of events [50]. A transition matrix might represent sequences of event types [39]. Furthermore, a variety of visualization approaches is available specifically

for analyzing eye movement data [7]. For interaction data, a sequence of thumbnails could represent the state of a user interface over time [21, 23]. Another approach is to visualize interactions as a glyph on a time axis [16, 30]. A triangulation of different data streams, however, may lead to more valid results [6, 30, 43]. Eye movements can be aligned with transcribed think-aloud data and visualized over time [26]. A combined analysis of transcripts and interaction logs might represent the data on a timeline [8] or as a state diagram [44]. Also, combinations of interactions and eye movement data on a timeline are possible [15, 25]. However, combining all three data types allows a more holistic analysis to answer questions about why a task is performed (transcript), how a task is performed (interaction logs), and what a task pertains to (eye movements) [9, 6]. Blascheck et al. [6] represent eye movement and interaction data on a timeline with transcript data available on demand. In contrary to these approaches, we focus on the transcript—aligned with eye movement and interaction data—as the main data stream because we consider the transcript most relevant for coding.

Our approach uses *word-sized visualizations*, which are also known as *sparklines* [55], to embed additional information into a user interface in a space-efficient way. This approach combines two ideas from previous work: First, it applies *word-sized eye movement visualizations* [3] for encoding eye movement and interaction data. Second, word-sized visualizations show overlap of codes as part of a faceted browsing concept, similar to the approach in *SurVis* [4] for bibliographic data. In general, word-sized visualizations have been explored for different purposes, for instance, enriching natural-language text [20], or as part of visualizations [37]. However, they have not been used for supporting analysts in coding user behavior or interactive editing scenarios.

For a comparison of coding categories, we introduce a visual set comparison approach. There exist many techniques to visualize set structures, as Alsallakh et al. [1] survey in a recent state-of-the-art report on set visualization. They describe comparing set families as an open problem. Among the surveyed techniques, *Parallel Sets* [34] comes closest to ours: they visualize different categorizations of elements on connected axes, but only allow non-overlapping partitions of elements on each axis; variants of this approach were also used in different contexts [46, 56]. Other visualization techniques look similar, but represent different data, for instance, multivariate data [28], dynamic graphs [11], or changing hierarchies [53]. Approaches, which support comparing different word clouds are also related: *Parallel Tag Clouds* [13] use parallel axes and *RadCloud* [10] integrates several word clouds into one radial diagram. Specifically for comparing assigned codes, MAXQDA combines a code relations browser, which shows codes on two axes of a matrix encoding their overlap in the cells [47].

## 3 CODING USER BEHAVIOR

For analyzing user behavior, we split the continuous stream of information into discrete, non-overlapping periods, which we call *user activities*. Each activity summarizes a small set of actions and has a precisely defined start time and duration. Hence, an activity specifies a segment of the recorded video, audio, eye movement, and interaction data. We assume that the process of identifying activities of meaningful granularity has been done as a preprocessing step. Further, every activity has a textual description attached, which is part of the experiment transcript. We suggest transcribing and identifying activities in one manual preprocessing step.

Categorizing and structuring user activities by assigning *codes* is the focus of this paper. We follow a common definition of *code*:

“A code in qualitative inquiry is most often a word or short phrase that symbolically assigns a substantive, salient, essence-capturing, and/or evocative attribute for a portion of language-based or visual data.” [47]

<sup>1</sup>[https://en.wikipedia.org/w/index.php?title=Computer-assisted\\_qualitative\\_data\\_analysis\\_software&oldid=702988839](https://en.wikipedia.org/w/index.php?title=Computer-assisted_qualitative_data_analysis_software&oldid=702988839)

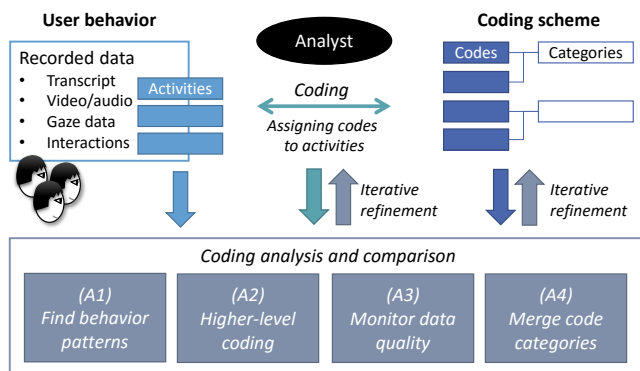


Figure 2: Interactive coding of user behavior, which our approach supports.

Hence, codes are simple textual descriptions of data and can be considered as categorical attributes of user activities. In other contexts, codes are called *keywords*, *tags*, or *terms*. To discern different types of codes, we introduce *code categories*: every code belongs to exactly one of these categories. We decided against using arbitrarily nested hierarchical categories to limit the complexity of editing codes and categories. Also, a hierarchy is just one option to further structure code categories: relationships between categories and codes might be more complex and need to be organized in a graph. We consider category organization and relationships out of scope in this work.

*Coding*—the process of assigning codes—is an important part of both qualitative and quantitative research methods. In general, there exist various coding styles structured by Saldana [47] into two cycles: after initial coding of entities in the first cycle, the second cycle builds upon first-cycle coding to further structure and abstract concepts. Our approach is generic and neither specialized nor limited to a specific style or cycle. In qualitative research, the popular *Grounded Theory* methodology [14, 19] heavily relies on coding. According to Corbin and Strauss [14], the process starts with *open coding* as an initial step, *axial coding* derives categories, and finally *selective coding* helps build a *theory* of coded concepts. *Grounded Theory* was already used to analyze user behavior, for instance, evaluating visualization systems [29, 40], or software development tools [36, 40]. Also, researchers applied parts of *Grounded Theory* coding methodology or an ad-hoc coding method to analyze qualitative user data [32, 44]. Moreover, coding is relevant in *mixed-methods research*—combining qualitative and quantitative methods—where codes make qualitative information *countable* and coding forms an intermediate step for a quantitative analysis [2, 51]. Evaluating visual systems according to the seven scenarios [35], coding of user behavior is particularly important for *understanding environments and work practices*, *evaluating visual data analysis and reasoning*, and *evaluating user experience*.

In particular, our approach supports the coding process illustrated in Figure 2. The behavior of multiple users summarized in a transcript, enriched with additional data, and split into activities is the starting point of the process. An analyst assigns codes to activities and, as part of this process, applies an existing coding scheme or develops a new one. However, beyond lower-level coding, we also want to support the following higher-level analysis tasks.

- **(A1) Find behavior patterns:** Summarizing activities assigned to specific codes, our approach facilitates exploring common behavior patterns as part of a first coding cycle, for instance, related to eye movements or interactions [6].
- **(A2) Higher-level coding:** Overlap of codes and similarity in patterns helps to abstract codes to higher-level codes, creating new categories as part of a second coding cycle [47].

- **(A3) Monitor data quality:** Analyzing user behavior involves automatic and manual processing steps, and is prone to mistakes and inconsistencies; an analysis of the recorded data and current coding could help fix data quality issues early.
- **(A4) Merge code categories:** Code categories might describe orthogonal dimensions or related concepts; in the latter case when several coders have done coding independently, code categories need to be consolidated and merged [47, 51]. This requires a comparison between coding results.

This analysis and comparison of coded data leads to an iterative refinement of both individual code assignments and the overarching coding scheme consisting of codes structured into categories. The outcomes of the process are not only assigned codes and a developed coding scheme, but also insights into behavior patterns and relationships between coded concepts.

The described coding process implemented in a visual interface facilitates the reasoning process of an analyst [54] in different ways: a visual interface integrates information sources in a single representation, allows assigning codes interactively based on observed data, and provides means to visually explore assigned codes. The coding process can also be mapped to the visual analytics process defined by Keim et al. [31]. While recorded user behavior split into activities represents the *data*, the *model* consists of codes assigned to user activities and a coding scheme, which an analyst develops during coding. The employed *visualization* techniques—word-sized visualizations and the comparison diagram—represent model and data. These representations help an analyst to gain *knowledge* in the form of theories resulting from qualitative methods or quantitative statistics, which summarize the coded user behavior.

Based on the analysis tasks, we compare different systems presented in the related work section with our approach. These include: the CAQDAS systems MAXQDA and ATLAS.ti as well as the research projects ChronoVis [57], SensePath [41], and the Operation Analysis Tool [16] (see Table 1). As features, we chose the support of different data sources, comparison of different variables, search and filtering options, as well as visual analysis support. We believe that these four categories are well-suited to describing the differences between the approaches. Overall, the two CAQDAS systems are most similar to our approach; however, they do not cover the same data sources we do. The three research projects are in comparison rather dissimilar to our approach. This is not surprising because the tools are specialized on other use cases. ChronoVis [57], among these, is closest to our approach, but ChronoVis supports a visual analysis of multi-modal data sources based focusing on the video as the main data source.

## 4 VISUAL CODING APPROACH

For visually supporting coding of user behavior, we designed a multiple-coordinated view [45] approach. Our approach consists of four non-overlapping panels organized in a predefined layout (Figure 1). We use brushing-and-linking [5] to connect the panels in a way that data edits and selections in one panel propagate to all panels. With this concept, we facilitate two general scenarios: First, *low-level coding* relies on the central *Selected Activities Panel* (Figure 1, B) integrating different data streams in a tabular representation and the *Sidebar* supports it, showing video and details on demand (Figure 1, C). Second, *high-level coding and analysis* facilitating the previously discussed analysis tasks (A1–A4) additionally requires the *Selection Panel* (Figure 1, A) for faceted browsing and the *Comparison Panel* (Figure 1, D) to analyze the coding scheme.

### 4.1 Data Input

Our approach assumes that eye movement data, interaction logs, and a video were recorded during a user study. Additionally, as the main data asset, the user behavior needs to be transcribed

Table 1: Feature matrix comparing different coding software systems and approaches. Legend: ● = supported, ○ = not supported.

Features	MAXQDA	ATLAS.ti	ChronoVis [57]	SensePath [41]	Ops. Anlys. [16]	Our approach
<b>Data sources</b>						
Transcript/text	●	●	●	○	○	●
Eye movement data	○	○	●	○	○	●
Interaction data	○	○	●	●	●	●
Audio recording	●	●	○	○	○	○
Video recording	●	●	●	●	●	●
<b>Comparison</b>						
Comparison of codes	●	●	○	○	○	●
Comparison across users	●	●	○	○	○	●
Comparison across researchers	●	●	○	○	○	●
Comparison across systems	○	○	○	○	○	●
<b>Search/filter features</b>						
Text search	●	●	○	○	○	●
Pattern search	○	●	○	○	○	●
Filter by category	●	●	●	○	○	●
Filter by code	●	●	●	○	○	●
Filter by user/document	●	●	○	○	○	●
<b>Visual analysis</b>						
Code relations/comparison	●	●	○	○	○	●
Frequency analysis	●	●	○	○	○	●
Code summary	●	●	○	○	○	●
Visualization of user behavior	●	●	●	●	●	●

(e.g., based on a thinking-aloud protocol) and split into activities. Recorded eye movements and interactions are treated as events having a timestamp. Each event can be located on the screen. The straightforward data structure makes our approach easily extendable to other data sources, as only a timestamp is required to synchronize different data streams. Dividing a stimulus into non-overlapping *Areas of Interest* (AOIs), we assign both types of events to an AOI as a categorical attribute. Although the examples shown in this work use rectangular static AOIs, our approach is open to non-rectangular or dynamically changing AOIs. To create eye movement events, we aggregate subsequent fixations in the same AOI into a single dwell accumulating individual fixation durations. For interactions, we allow assignment of an interaction type. In the following, we use the eleven categories defined by Brehmer and Munzner [9] as an example: *encode*, *select*, *navigate*, *arrange*, *change*, *filter*, *aggregate*, *annotate*, *import*, *derive*, and *record* (cf. Fig. 1, C, *Legend*). While transcribing and identifying activities are manual processes, the other steps—recording, synchronizing, and preprocessing—can be automated.

The dataset used in the paper originated from a study in which the visual analytics system *VarifocalReader* [33] was evaluated. *VarifocalReader* is a system for analyzing large documents on different levels (i.e., chapter, subchapter, page, paragraph, textpage).

## 4.2 User Activities

The *Selected Activities Panel* (Figure 1, B) is the main panel of our approach where the coding process takes place. It contains a visually enriched table of currently selected user activities. The tabular representation allows an integration of data with multiple data types using embedded visualizations. Furthermore, a table can be easily searched, filtered, or reordered. Each row represents an individual activity of a specific user. The table might list activities of the same user or multiple users in several rows. An activity contains an activity ID, a start timestamp, a duration, a textual transcription, a

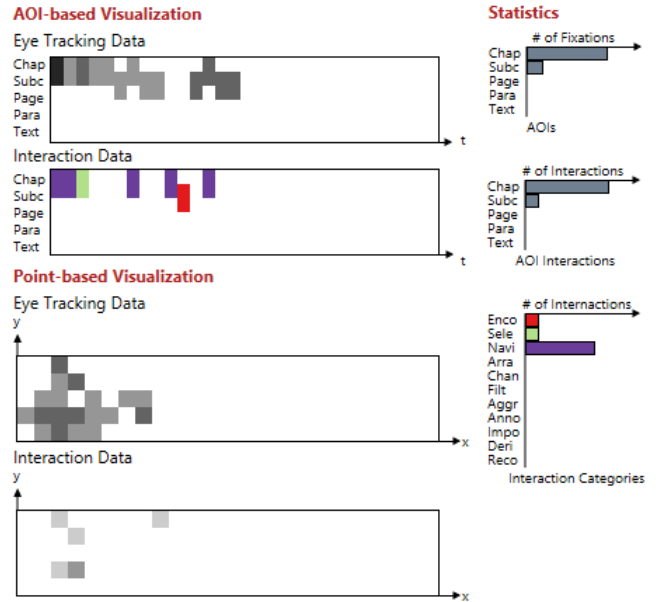




Figure 3: Details of a selected activity including enlarged and labeled word-sized visualizations and additional statistics in histograms. The color coding, both in the word-sized visualizations and in the histograms, refers to the interaction categories. The gray-scale values in the word-sized visualizations represent the AOI dwell times for the AOI-based visualization as well as the fixation duration and interaction count for the point-based visualizations.

thumbnail image, word-sized visualizations for eye movements and interaction data, and a column for each coding category. The category columns allow an analyst to immediately add existing or new codes to an activity using an auto-complete text box. The *Selection Panel* (Figure 1, A) displays the current coding scheme below a list of available users. To refine a coding scheme, categories and codes can be created, edited, and removed using a context menu.

In the activity table, thumbnail images display the state of the screen at the start time of an activity, helping an analyst to quickly recall the state of an interface [23]. A subsection of an activity shows details about this activity in the *Sidebar* (Figure 1, A). It contains a video playback, an enlarged version of the word-sized visualizations, and histograms. We chose to integrate a video playback option into our approach to recall actions and review video segments for ambiguous activities [16]. The video plays at the start time of a selected activity. It can be overlaid with additional information. In our examples, we use a visualization of eye movements and interactions, which show an animated scanpath and mouse click events. In addition, details in the *Sidebar* (Figure 3) depict three types of histograms, which show (i) the number of fixations per AOI, (ii) AOIs of each interaction and the interaction count per AOI, and (iii) interaction count per interaction category.

The word-sized visualizations depicted in the *AOI-based vis* and *point-based vis* column are adapted from the concept of *word-sized eye tracking visualizations* [3]. These visualizations integrated into the activity table enable an analyst to code user activities without switching between multiple views or windows. On subsection, word-sized visualizations are shown enlarged in the *Sidebar* with labels (Figure 3). The AOI-based word-sized visualizations show eye movements  and interaction events  as rectangles on a timeline from left to right; the vertical position of each rectangle represents the AOI an event belongs to. The rows in the visualization overlap to increase the area of the rectangles and the discernibility of colors. The color coding of eye movements corresponds to the dwell time (Figure 1, C, *Legend*), which describes

the time a user has focused on a specific AOI. The color coding of interactions corresponds to the interaction categories. We use a qualitative color table created with *ColorBrewer* [22] having eleven distinct colors. The point-based word-sized visualizations display eye movements and interactions in a representation similar to an attention map. In this case, the visualization is divided into a grid of rectangles, where each rectangle covers a part of the recorded screen area. For each rectangle, fixation duration or interaction count is accumulated and then color-coded in the rectangle (Figure 1, C, *Legend*). An analyst might use the word-sized visualizations during a coding process to visually search for activities, which have similar eye movement or interaction patterns. These visualizations can be considered as an aligned small multiples representation of this data across the rows of the table. There are other options to design word-sized visualizations for eye tracking and interaction data, which we have not implemented yet [3], but would be easy to integrate.

### 4.3 Faceted Browsing

We use faceted browsing [58] to filter and search for user activities. As listed in the *Selection Panel* (Figure 1, A), available facets—the criteria for selecting activities—are users, categories, text search, and pattern search. Every query yields a set of activities  $S$  shown in the *Selected Activities Panel* (Figure 1, A); the heading indicates the number of currently selected activities  $|S|$ . Multi-selection of facet values is possible for users and codes of the same category using *Ctrl*-click, which increases the number of results in  $S$  applying an **OR** operation. For example, for two users with activities  $S_1$  and  $S_2$ , the selected activities are  $S = S_1 \cup S_2$  when both users are selected. It is also possible to select all users with a button. Selecting a code category by clicking on it is equivalent to selecting all codes of a category together. In contrast, if values from different facets are selected (e.g., a user with activities  $S_1$  and a code with activities  $S_2$ ) activities are restricted based on an **AND** operation ( $S = S_1 \cap S_2$  only contains activities of a user that have a selected code). For rapid coding, an option to collectively add or remove a code to or from all currently selected activities is available in a context menu.

To show the degree of congruence of a current selection with the recorded users and assigned codes, we add further word-sized selection visualizations to the *Selection Panel* (Figure 1, A). For each category and code, a small bar chart represents the fraction of currently selected activities, which have assigned this code or any code in the category respectively. For example, if 15 of 20 currently selected activities contain a code, the rectangle next to the code is filled to three quarters. A word-sized selection visualization is also depicted next to a user . This visualization represents the temporal order of selected activities with respect to all activities of a user. Occurrences of selected activities are indicated by filled rectangles on this timeline. In addition, a subselected activity is highlighted in brighter blue. These visualizations are helpful, for example, for seeing if a current selection refers to the beginning, middle, or end of the user’s activities.

Searching for words in the transcripts or finding specific patterns in the event data can help an analyst to speed up the coding process. Specific words or patterns may indicate that specific codes should be assigned. Entering a text string into the search box selects those activities containing this string in the transcript and highlights it. Our pattern search is based on a pattern editor to identify specific sequences of eye movement and interaction events (Figure 4a) [6]. Each stack of rectangles represents an element of an event sequence and each rectangle identifies an AOI. A filled gray rectangle selects an eye movement event for a search while a color-coded circle is used to search for interactions in a specific AOI with a chosen interaction type. An analyst can create a search pattern by adding individual pattern elements. Wild card elements can be included to represent arbitrary events. In the exact search mode, an event has

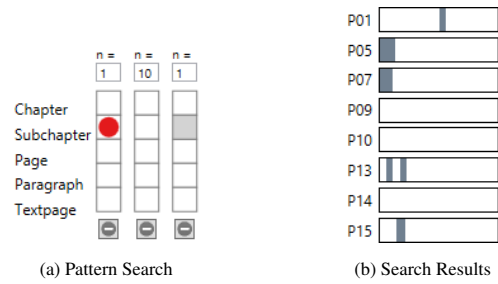


Figure 4: (a) An analyst can create search patterns in the pattern editor. For example, a fuzzy search pattern finds activities which contain an *encode* interaction (red circle) followed by an investigation of the *Subchapter* AOI (gray rectangle). (b) The search results depict all occurrences of the created search pattern. They are shown as word-sized visualizations next to the user labels.

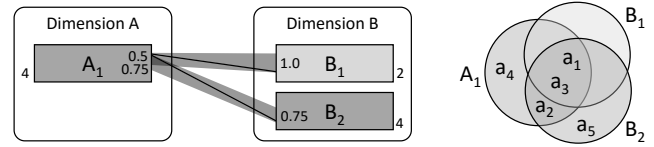


Figure 5: Illustration of the comparison visualization (left) showing three activity sets in two dimensions (numbers indicate the size of activity sets and similarity values of edges) and traditional Venn diagram representation of the sets (right).

to occur exactly as often as listed in the top row. In contrast, the fuzzy mode interprets these numbers as maximum occurrences. In the example in Figure 4a, a fuzzy search specifies a pattern where users first performed an *encode* interaction in the subchapter AOI and, up to ten events later, looked at this AOI. All activities containing this pattern are selected and the pattern is highlighted in the AOI-based word-sized visualization . In the user sparklines the occurrences of a pattern are shown as well.

### 4.4 Comparison Visualization

We extend browsing and analyzing individual activities using a dedicated comparison view (Figure 1, D) to facilitate higher-level analysis of assigned codes. We want to analyze shared activities of codes in a single category, compare different code categories, and put code categories in context of other categorical attributes. These additional attributes include users who performed an activity and AOIs that users looked at or interacted with. While further predefined attributes would be possible, the faceted browsing in combination with collective assignment of codes provides a flexible way to introduce new dimensions to the comparison by creating new code categories on demand. In general, comparing codes and categorical attributes helps refine and abstract assigned codes and the coding scheme (cf. Figure 2, A2–A4).

Visually comparing categorical attributes involves set comparison because attributes form sets of elements. In particular, a specific code  $c$  unambiguously identifies a set of user activities  $A_c$  equivalent to selecting a code. Since we allow the assignment of multiple codes per activity, even for the same category, those sets of activities might overlap. A code category with  $n$  codes, can be described as a family of sets  $A = \{A_1, A_2, \dots, A_n\}$  where each  $A_i$  represents a set of activities. Analogously, we model other categorical attributes as families of activity sets. A division into users, however, forms a partition of the set of all recorded activities (i.e., every activity is assigned to exactly one attribute value). As a generalization, we refer to assigned codes and categorical attributes as *dimensions*.

Figure 5 illustrates the design of the comparison diagram (Figure 1, D) and shows a comparison of two dimensions  $A, B$ . As in *Parallel Coordinates* [28], vertical axes represent dimensions, here

as rounded rectangles. This allows adding multiple dimensions at the same time, while a matrix representation would only allow the comparison of two dimensions. Dimensions can be selected from a list at the left side of the *Comparison Panel*. A direct comparison is performed between neighboring dimensions. Adding a dimension multiple times allows an analyst to compare a dimension with itself (e.g., to analyze the overlap of codes) or with several other dimensions. A dimension  $A$  contains all available codes or categorical attribute values  $\{A_1, A_2, \dots, A_n\}$  as nodes, each representing a set of activities. We encode the number of activities per set in the background of nodes, mapping the size logarithmically to a gray scale—the larger, the darker. We refrain from coloring the nodes because color is used for other parts of the system. In this case, gray-scale coding is sufficient to represent the number of activities.

To support visual comparison of dimensions, every two nodes  $A_i, B_j$  of neighboring dimensions  $A, B$  are linked based on their similarity  $\text{sim}(A_i, B_j)$  if  $\text{sim}(A_i, B_j) > 0$ . We decided to use an asymmetric similarity function (i.e.,  $\text{sim}(A_i, B_j) = s \not\Rightarrow \text{sim}(B_j, A_i) = s$ ) to better reflect the direction of subset relationships. As an asymmetric variant of the symmetric *Jaccard coefficient* and similar to *precision/recall rates*, we define the following similarity function:

$$\text{sim}(A_i, B_j) = \frac{|A_i \cap B_j|}{|A_i|} .$$

If we compare a set of activities  $A_i$  with itself, similarity reaches its maximum of 1 ( $\text{sim}(A_i, A_i) = 1$ ). However, in contrast to the *Jaccard* coefficient, we also get a maximum for comparing  $A_i$  to  $B_j$  if  $A_i$  is a subset of  $B_j$  ( $A_i \subset B_j \Rightarrow \text{sim}(A_i, B_j) = 1$ ). Since the function is asymmetric, comparing  $B_j$  to  $A_i$  (assuming that  $A_i$  and  $B_j$  are not equal) does not produce a maximum similarity ( $A_i \subset B_j, A_i \neq B_j \Rightarrow \text{sim}(B_j, A_i) < 1$ ). If two compared sets are disjoint, the function is always 0 ( $A_i \cap B_j = \emptyset \Rightarrow \text{sim}(A_i, B_j) = 0$ ).

To visualize the similarity of two nodes, we map the similarity value to the thickness of edges connecting nodes. Since our employed similarity function is asymmetric, we need to use directed edges. Tapered links [27]—representing an edge as a triangle or similar pointed shape—are a space-efficient and readable way to encode direction information. Illustrated in Figure 5, we place two triangles representing the two directions next to each other, with a thin line separating the two. If the similarity values are equal, the bars form a rectangle. Assuming we compare  $A_i$  on the left to  $B_j$  on the right, the lower triangle encodes  $\text{sim}(A_i, B_j)$  and the upper one encodes  $\text{sim}(B_j, A_i)$ . Hence, if both similarity values are different, the bar becomes asymmetric: it is larger on the one end than the other. Hovering a triangle shape allows one to explore the links and a detailed tooltip dialog explains how metric values are computed.

Nodes can be sorted to reduce edge crossings between adjacent leaves, either manually via drag-and-drop or with an automatic method. We use the Sugiyama layout for hierarchical graphs [52] as an automatic method, sweep through all dimensions, and optimize the current dimension in relation to the previously optimized dimension. In particular, we use the median method [17] as a heuristic to calculate node positions in each optimization step.

The comparison diagram is linked with a currently selected activity. That means the numbers and similarity values visualized always refer to activities currently shown in the *Selected Activities Panel* (Figure 1, C). By changing the selection, an analyst might perform different analyses, for instance, investigate the coding of a single user or related to a specific eye movement pattern. Dimensions and nodes in the comparison diagram act as selection inputs as well: clicking on a node representing a code or user is equivalent to selecting a code or user in the *Selection Panel* (Figure 1, A).

## 5 CASE STUDY

To demonstrate our approach, we present a case study analyzing data from a user study [6], where we collected think-aloud, eye

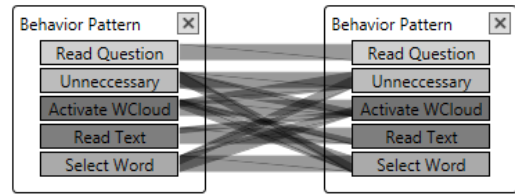


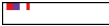
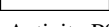


Figure 6: Comparison diagram showing the overlap of coded behavior patterns of P01.

movement, and interaction data. The study evaluates *Varifocal-Reader* [33], a visual analytics system which uses a smooth hierarchical scroll navigation to explore large text documents. Different parts of a document (i.e., chapter, subchapter, page, paragraph, textpage) are displayed in vertical layers. Each layer provides visualization techniques such as word clouds, bar charts, and pictograms. We recorded eye movement data using a *Tobii T60XL* eye tracker (60 Hz recording rate, 24-inch screen, 1920×1200 px screen resolution, and nine-point calibration). Gaze points are automatically clustered into fixations using the *Tobii Fixation Filter* (velocity threshold of 35 px/samples and distance threshold of 35 px). We instrumented the analyzed system to log interaction data. The think-aloud of users was recorded together with a screencast. Then, student research assistants transcribed the audio.

The structure of this case study follows the four higher-level analysis tasks defined in Section 3 and provides examples for each of them. In the following, we analyze eight users who participated in the study working on one task, leading to 95 activities in total.

### (A1) Find Behavior Patterns

The users in the study had to find a specific text fragment. To solve this task in an efficient manner with *VarifocalReader*, users need to activate a word cloud (*encode* activity), select one or multiple words in the word cloud (*select* activity), navigate to the appropriate text passage, and read small parts of the text. These are the specific behavior patterns, which we want to identify in the following. We first look for indicators in the eye movements and interaction logs displayed as word-sized visualizations. Then, we check the transcript and, if necessary, the video playback of an activity.

We start our investigation by analyzing one user in detail (P01). The transcript and video of the first activity show that P01 first reads the question describing the current task (*read question* code). Next, a user, ideally, activates the word cloud, indicated by a red rectangle (*encode* interaction) in the word-sized AOI-based visualization for interactions. However, it takes some time until P01 performs this first *encode* interaction (Activity P01\_04 ). From the visualization and transcript, it is still unclear if the word cloud was activated (there are other *encode* interactions). The video reveals that the user enabled a different visualization. We code this activity *unnecessary* for the task. This code can later be used to investigate problems with the user interface. Finally, in Activity P01\_09, the thumbnail image embedded in the table shows that P01 turned on the correct word cloud; we code this as *activate word cloud*. In Activity P01\_10, the user selects a specific word (*select word* code). Thus, we inspect the AOI where the word cloud was enabled (first row in Activity P01\_10 ) and a select activity which follows (green rectangle in Activity P01\_10 ). The last part of the task involves reading text to find and understand the relevant text passage. This requires that the user focuses on the textpage AOI in the right part of the interface (e.g., Activity P01\_16 ). We assign a *read text* code to activities following this pattern.

During the coding process, we attach multiple codes to one activity if several of the patterns were performed together. The comparison diagram visualizes this overlap when comparing the respective code category to itself. Figure 6 gives an example for the behav-

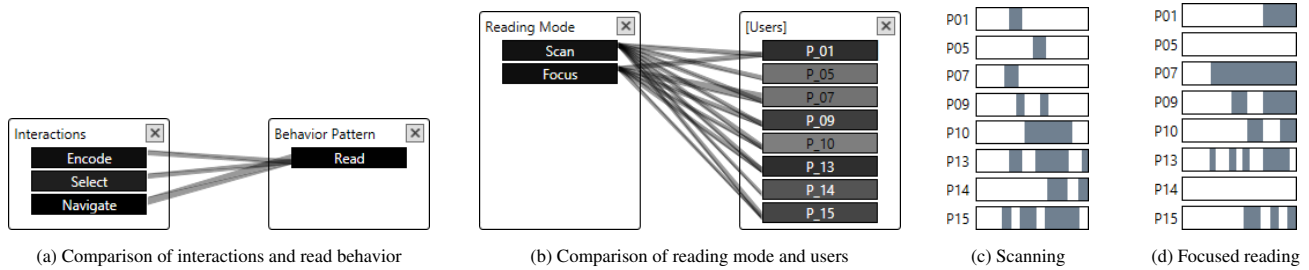


Figure 7: Analyzing reading behavior of users in detail. (a) After a first coding cycle, comparing interaction codes with reading. (b) After a second coding cycle, comparing scanning and focused reading to users. (c) Identified scanning activities for all users in user timelines. (d) Identified focused reading activities for all users in user timelines.

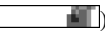
ior pattern codes assigned to the activities of P01. For instance, *read question* never occurs together with one of the other codes. However, *activate word cloud* is often executed together with *select word*. Specifically, in the activities coded with *select word*, P01 always performs an *activate word cloud* action, but only half of the *activate word cloud* activities carry a *select word* code:

$$\text{sim}(\text{select word}, \text{activate word cloud}) = 1.0 \quad ,$$

$$\text{sim}(\text{activate word cloud}, \text{select word}) = 0.5 \quad .$$

Next, we want to extend the coding to all users. As an example, we show how to find occurrences of *activate word cloud*. To this end, we create different fuzzy search patterns. In the example depicted in Figure 4a, the search pattern retrieves all occurrences where a user performed an *encode* interaction on the subchapter AOI and later visually investigates this AOI. Figure 4b shows that this pattern occurs, as expected, at the beginning of the experiment (as noted earlier, P01 forms an exception). To make sure that these activities are *encode* interactions we are interested in, we watched the videos of the results before coding the activities with *activate word clouds*. Since this pattern did not yet cover the *activate word cloud* activities of all users (cf. Figure 4b), we repeat this search with a similar pattern where users activated the word cloud in the chapter AOI. Afterwards, we search for activities where users selected a word again for the chapter and subchapter AOI and classify the resulting activities as *select word*.

### (A2) Higher-Level Coding

As part of a second coding cycle, we want to abstract codes to higher level ones. In a first coding cycle, we coded user activities using the point-based word-sized eye movement visualizations. We visually inspected the visualizations and looked for large clusters in the right part (e.g., Activity P01\_07 ) to identify reading behavior. In addition, using the pattern search, we coded occurrences of certain interactions of all users. Comparing these to the reading activities as shown in Figure 7a, we see a strong relation between reading and navigating—a user might not just read but also quickly scroll through a document. Having a closer look at the reading activities, we found that users actually applied different reading strategies. We were able to differentiate the read activities *scanning* and *focused reading* into a new code category. Inspecting this new dimension in the *Comparison Panel* contrasting the *reading mode* with *users* as shown in Figure 7b indicates three groups: users applying only scanning (P05 and P14), users applying almost exclusively focused reading (P07) (cf. Fig. 7d), and users applying both scanning and focused reading (all other users). This result indicates that the task users performed could be solved with little or no focused reading.

As described above (A1), we coded all occurrences of *activate word cloud* and *read text*. Next, we coded all occurrences of the word *Wallenstein*, which is an important term to solve the task, using the text search. Selecting the code *Wallenstein*, we found that,

at the beginning, this code is mostly associated with *activate word cloud*, and at the end, it mostly appears with *read text*. Thus, we create two new codes and assign them to the activities.

### (A3) Monitor Data Quality

Monitoring data quality is an important part when evaluating a user study. Mistakes can happen in different parts of a study. Users might misunderstand a task or somehow invalidate the recorded data. For example, P05 accidentally found the solution of the task without once mentioning the word *Wallenstein*. We identified this outlier using the comparison diagram to compare this code with all users. Furthermore, a typical problem during the study was that users did not adhere to thinking aloud and had to be reminded. Thus, some of the transcripts are rather sparse. In the activity table, this becomes immediately obvious when comparing the length of a transcript of an activity to the activity duration or amount of visualized events—other data streams fortunately still provide some data to interpret and code activities. In some cases, code assignment is ambiguous because different criteria exist. For example, we assigned *read* to all activities that had a high amount of attention on the right part of the interface as discussed earlier. When refining this coding with the higher-level codes *scan* and *focus*, the video showed that, although there was a lot of attention on the right part of the interface, some activities are not reading activities. In general, the integration of different data streams often prevented us from misinterpreting the data. It is simple to double-check and refine the coding using a second data source.

### (A4) Merge Code Categories

To show how our approach can be used to merge code categories, two of the authors independently coded data from two users (P07 and P14) using an existing coding scheme. In visualization research, an approach to evaluate systems is to code and analyze insights users gain from the data while using a visualization. To this end, Saraiya et al. [49] propose a coding scheme containing the categories *overview*, *pattern*, *group*, and *detail*. However, *group* and *pattern* were not applicable in our scenario. Including the extensions by Smuc et al. [50], we added the categories *data insight* and *tool insight*. Coder C1 needed 17 minutes to code the two users with 19 activities and coder C2 12 minutes. The typical approach both authors used was to first look at the video and do a preliminary coding and then do a refinement mainly using the transcript, thumbnails, and word-sized visualizations only. We used two code categories to discern the codes of C1 and C2.

To compare the two codings, we loaded them into the comparison diagram (Figure 8a). The diagram shows some overlapping edges, which indicates that the two codings differ. For example, Coder C1 did not use the code *Tool Insight*, but assigned the codes *Detail* and *Overview* more often. Both coders applied the code *Data Insight* similarly. The code *Overview* of C2 is a subset of *Overview* of C1. Next, both coders discussed the individual decisions and

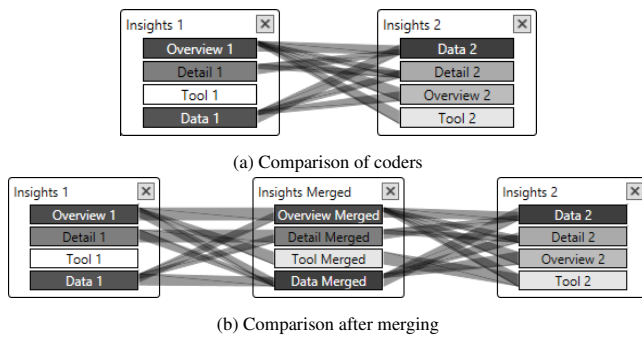


Figure 8: Comparison diagram of insights two coders coded. (a) Comparison of codings of the two coders before merge. (b) Comparison of codings of two coders with merged codes in the middle.

merged codes creating a new category. Figure 8b shows a comparison with the merged category in the middle. Here, we find that the codes *Overview* and *Detail* were taken from coder C1, whereas the codes *Tool Insight* and *Data Insight* were mostly taken from coder C2. Although both coders used the same coding scheme and discussed the meaning of the individual codes beforehand, we realized that a consistent coding is hard to achieve and a comparison is necessary to finalize a coding. The comparison diagram eases this process and shows differences and changes.

## 6 FIRST USER FEEDBACK

To complement the case study, we reached out to colleagues who are either experts in visualization, human-computer interaction, or user studies. We sent them an email invitation with a link to a demonstration video, accompanied by a short questionnaire containing four closed-ended and five open-ended questions. This preliminary user feedback aimed at collecting high-level comments on strengths and weaknesses of our general approach as well as ideas for future extensions. We provide the demonstration video in a slightly revised version as well as the questionnaire as supplementary material. Participants were asked to rate their expertise in the areas of visualization, human-computer interaction, and user studies; possible answers were *no knowledge*, *passing knowledge*, *knowledgeable*, and *expert*. In the analysis, we included only participants who rated themselves in at least one of these areas as *experts* or as *knowledgeable* in at least two areas. Further, we asked for experts' experience with relevant research methods including, for instance, coding qualitative data, think-aloud, and eye tracking. We excluded respondents who did not apply at least two of these methods in the past. From 19 responses we received, we had to omit two which did not match the above criteria. We analyzed and coded the open-ended answers and divided them into positive and negative feedback. In the following, we refer to individual experts using E1–E19. We also provide the codes along with the corresponding statements as supplementary material.

The most positive feedback were statements highlighting the integration of various data sources in our approach, expressed by ten experts. E9, for instance, reported that he or she usually employs different tools like Excel, mind maps, or custom tools which are not linked, and further states that “[a]ccessing everything from one tool seems to make analyzing different data sources so much easier.” Also, six experts highlighted the filtering capabilities of our approach. E7 wrote that this feature “could be very useful to identify patterns in user studies.” Besides feedback for particular panels of the GUI, five experts pointed out the good overview which the approach provides for codes, categories, and eye movement data. Further, four experts emphasized the brushing-and-linking approach and the resulting real-time updates of visualizations. E2, for example, emphasized the possibility to “refine different aspects [...] to see real-time effects on whole data processing stream.”

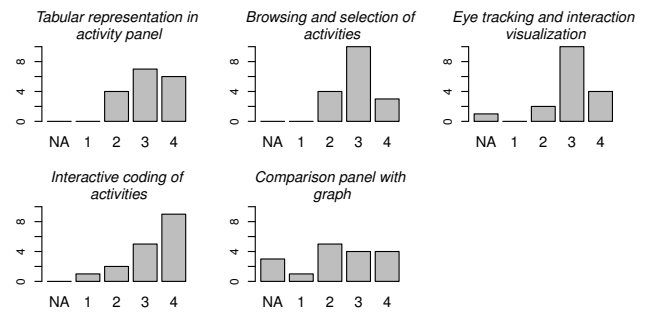


Figure 9: Experts' ratings of the potential of certain features on a 4-point semantic differential scale from *not promising at all* (1) to *very promising* (4), NA=*I don't know*; n=17.

Most doubts referred to the complexity of the user interface. Seven participants commented on this aspect, but two of them also provided suggestions about how to reduce this complexity, for example, by making the GUI more modular. E2 suggested to “have separate screens for the different analysis tools accessible via a menu” as this “will free up space, and allow [the user] to improve the clarity of some of the existing features/visualizations (e.g., point-based visualization) and continue to add new elements (e.g., new statistics plots).” Six experts commented on particular visualizations like the one used in the *Comparison Panel* or they suggested new visualizations like “[p]ictogram-based encoding of interactions” (E6) or “[v]isualizations that combine some of the various data sources” (E1). Aggregation may not only be utilized for different data sources but also for multiple subjects, as E16 suggests. Other possible improvements are related to particular details of the GUI like the legend, the timestamps, or the video playback. Regarding the video playback, E15 proposed to “show the current playback position in the two AOI-based [...] graphs in the details panel.” E17 made a similar remark.

In one of the closed-ended questions, we asked experts to rate the potential of five features on a 4-point semantic differential scale [24] from *not promising at all* (1) to *very promising* (4). Figure 9 shows ratings for each feature as bar charts. Four features were rated as relatively promising, where the system's capability to interactively code activities received the highest ratings. One feature, the *Comparison Panel*, was rated relatively low. The fact that three experts answered “I don't know” for that feature suggests that the explanation in the demonstration video was not clear enough. The open-ended answers also supported this impression: E9, for instance, noted that “the *Comparison Panel* seemed to be not as self-explaining as most of the UI”. As that part of our approach is novel and rather complex, analysts might need some time to familiarize themselves with it.

The generalizability of the expert feedback is limited due to the fact that we only reached out to experts we knew. Moreover, the feedback is based on a demonstration video and only two experts had prior experience with other CAQDAS systems. Still, experts' responses already supported the development of our approach as we incorporated part of the feedback and fixed usability issues in the final version of our approach. We plan to further evaluate our approach in a more thorough study with a larger group of users and under controlled conditions in order to obtain additional feedback as well as an objective and quantitative assessment.

## 7 DISCUSSION AND FUTURE WORK

The case study and expert feedback provide initial evidence that our approach is valuable for analyzing qualitative data of user behavior. In particular, the seamless integration of different data streams shows high potential to improve coding quality and efficiency significantly. The good overview of all data streams, which the activity



table provides, allows analysts to quickly identify and code behavior patterns. Data quality issues of different kinds stand out as outliers in the visualizations or through the comparison of data streams. Faceted browsing based on codes and flexible pattern search combined with linked visualizations support a higher-level analysis and second-cycle coding. The novel comparison diagram is a versatile tool to visually analyze a coding. However, our experts rated it not as promising as other features of our approach—an explanation might be that it is somewhat more complex and not fully self-explaining. We hypothesize that an analyst will only be able to use the full potential of this representation after working with the approach for several hours. However, there is potential for analysts to quickly learn how to use the system to code and establish a coding scheme for comparison.

A limitation of our approach is that we assume that user behavior can be split into discrete activities. This step is essential for implementing a faceted browsing approach, which requires distinct entities. In our solution, specific codes can be used to group low-level activities to larger, evenly-gapped sequences of high-level activities. We could extend our approach by allowing to split and merge activities within the system. To realize overlapping activities, parts of the data might be duplicated and be assigned to several activities. However, when merging codings from several coders, coders still need to agree on a specific partition of user behavior into activities first, otherwise assigned codes cannot be directly compared with each other. It would be interesting to explore different automatic and semi-automatic approaches for activity identification to simplify the required preprocessing steps.

While coding assigns entities a categorical attribute, *quantitizing* [48] describes a general process of assigning numerical attributes. Although our approach does not target this scenario explicitly, interpreting categorical attributes as numeric (e.g., codes such as “5”) could realize it. Assigning numeric attributes provides further opportunities to visually analyze coded data, which our approach does not yet exploit. One opportunity would be to visually represent coded numeric data as further word-sized representations within the activity table.

Currently, analysts assign a discrete code to activities and, if multiple analysts code the same data, they have to agree on a coding scheme. Another option is fuzzy coding [12]. Different saturations could be used to indicate the degree of agreement between analysts. For example, the more analysts agree on a code, the darker a code is colored. Filtering for specific codes could retrieve a table where activities are ranked based on the level of agreement [4].

So far, our approach focuses on an analysis of a single user interacting with a visual interface. However, there exist scenarios where multiple users perform tasks together, for instance, pair programming or working with tabletop displays. Although we cannot explicitly represent multiple activities performed in parallel, our approach is open for assigning an activity to a group of users instead. This is feasible when users closely collaborate on a task. Including several eye movement or interactions streams would be possible, it just adds further columns to the activity table. Also, having different columns for the transcript might be desirable to discern different users or a user and an experimenter.

After using our tool, data evaluation might not yet be completed. For instance, coded activities could be used as additional categories in other eye movement visualizations [7]. When applying *Grounded Theory*, codes might form the basis to develop a *theory* as the result of an analysis; an analyst sets codes into relation and formulates *memos*. In a mixed-method approach, a quantitative evaluation might follow the coding process. These activities could be performed in an iterative process combining our approach with other evaluation methods.

## 8 CONCLUSION

We have presented a visual analytics approach, which supports coding of user behavior as part of qualitative or mixed-method user evaluations of interactive interfaces. In comparison to previous work and existing tools, our approach integrates data-rich recordings of user behavior into the coding process. Word-sized visualizations and a dedicated set visualization combined with faceted browsing build an approach, which is both flexible and easy to use. Our approach facilitates an analyst in addressing data analysis tasks as part of a coding process to generate a high-quality coding scheme of observed user behavior, which we demonstrated in a case study. 17 experts provided feedback and highlighted the importance of data integration in a consistent representation. While these results show that our suggested approach is already a valuable tool for evaluating user behavior, we plan to extend it to cover an even wider range of use cases and analysis scenarios.

## ACKNOWLEDGEMENTS

This work was funded by the German Research Foundation (DFG) as part of the Priority Program SFB/Transregio 161. Fabian Beck is indebted to the Baden-Württemberg Stiftung for the financial support of this research project within the Postdoctoral Fellowship for Leading Early Career Researchers.

## REFERENCES

- [1] B. Alsallakh, L. Micallef, W. Aigner, H. Hauser, S. Miksch, and P. Rodgers. The state-of-the-art of set visualization. *Computer Graphics Forum*, 35(1):234–260, 2016.
- [2] P. Bazeley. Computer-assisted integration of mixed methods data sources and analyses. In A. Tashakkori and C. Teddlie, editors, *SAGE Handbook of Mixed Methods in Social & Behavioral Research*, volume 2, chapter 18, pages 431–467. SAGE Publications, 2010.
- [3] F. Beck, T. Blascheck, T. Ertl, and D. Weiskopf. Exploring word-sized graphics for visualizing eye tracking data within transcribed experiment recordings. In *Proceedings of the First Workshop on Eye Tracking and Visualization*, 2015.
- [4] F. Beck, S. Koch, and D. Weiskopf. Visual analysis and dissemination of scientific literature collections with SurVis. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):180–189, 2016.
- [5] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.
- [6] T. Blascheck, M. John, S. Koch, L. Bruder, and T. Ertl. Triangulating user behavior using eye movement, interaction, and think aloud data. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 175–182, 2016.
- [7] T. Blascheck, K. Kurzhals, M. Raschke, M. Burch, D. Weiskopf, and T. Ertl. State-of-the-art of visualization for eye tracking data. In *EuroVis - STARs*, pages 63–82, 2014.
- [8] I. Boyandin, E. Bertini, and D. Lalanne. A qualitative study on the exploration of temporal changes in flow maps with animation and small-multiples. *Computer Graphics Forum*, 31(3.2):1005–1014, 2012.
- [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.
- [10] M. Burch, S. Lohmann, F. Beck, N. Rodriguez, L. Di Silvestro, and D. Weiskopf. RadCloud: Visualizing multiple texts with merged word clouds. In *Proceedings of the International Conference on Information Visualisation*, pages 108–113, 2014.
- [11] M. Burch, C. Vehlou, F. Beck, S. Diehl, and D. Weiskopf. Parallel Edge Splatting for scalable dynamic graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2344–2353, 2011.
- [12] F. Chevene, S. Doleadec, and D. Chessel. A fuzzy coding approach for the analysis of long-term ecological data. *Freshwater Biology*, 31(3):295–309, 1994.
- [13] C. Collins, F. B. Viegas, and M. Wattenberg. Parallel Tag Clouds to explore and analyze faceted text corpora. In *IEEE Symposium on Visual Analytics Science and Technology*, pages 91–98, 2009.

- [14] J. M. Corbin and A. Strauss. Grounded Theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, 1990.
- [15] E. C. Crowe and N. H. Narayanan. Comparing interfaces based on what users watch and do. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 29–36, 2000.
- [16] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. Lipford, and R. Chang. Recovering reasoning processes from user interactions. *IEEE Transactions on Computer Graphics and Applications*, 29(3):52–61, 2009.
- [17] P. Eades and N. C. Wormald. Edge crossings in drawings of bipartite graphs. *Algorithmica*, 11(4):379–403, 1994.
- [18] S. Friese. Computer-aided qualitative data analysis: an overview. In J. Fikfak, A. Frane, and D. Garz, editors, *Qualitative Research*, chapter 12, pages 199–229. ZRC Publishing, 2004.
- [19] B. G. Glaser and A. L. Strauss. *The discovery of Grounded Theory: Strategies for qualitative research*. Transaction Publishers, 2009.
- [20] P. Goffin, W. Willett, A. Bezerianos, and P. Isenberg. Exploring the effect of word-scale visualizations on reading behavior. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1827–1832, 2015.
- [21] D. Gotz and M. X. Zhou. Characterizing users’ visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.
- [22] M. Harrower and C. A. Brewer. ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.
- [23] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1189–1196, 2008.
- [24] D. R. Heise. The semantic differential and attitude research. In G. F. Summers, editor, *Attitude measurement*, chapter 14, pages 235–253. Rand McNally, 1970.
- [25] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, and J. Van de Weijer. *Eye Tracking: A Comprehensive Guide to Methods and Measures*. Oxford University Press, 1st edition, 2011.
- [26] J. Holsanova. Dynamics of picture viewing and picture description. In L. Albertazzi, editor, *Visual Thought: The Depictive Space of Perception*, pages 235–256. John Benjamins Publishing Company, 2006.
- [27] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2299–2308, 2009.
- [28] A. Inselberg and B. Dimsdale. Parallel Coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st Conference on Visualization*, pages 361–378, 1990.
- [29] P. Isenberg, T. Zuk, C. Collins, and S. Carpendale. Grounded evaluation of information visualizations. In *Proceedings of the BELIV Workshop: Beyond Time and Errors – Novel Evaluation Methods for Visualization*, pages 6:1–6:8, 2008.
- [30] M. R. Jakobsen and K. Hornbæk. Fisheyes in the field: Using method triangulation to study the adoption and use of a source code visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1579–1588, 2009.
- [31] D. Keim, G. Andrienko, J.-D. Fekete, C. Görg, J. Kohlhammer, and G. Melançon. Visual Analytics: Definition, process, and challenges. In *Information Visualization*, volume 4950, pages 154–175. Springer, 2008.
- [32] K. Kinley, D. Tjondronegoro, H. Partridge, and S. Edwards. Modeling users’ web search behavior and their cognitive styles. *Journal of the Association for Information Science and Technology*, 65(6):1107–1123, 2014.
- [33] S. Koch, M. John, M. Wörner, A. Müller, and T. Ertl. VarifocalReader – in-depth visual analysis of large text documents. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1723–1732, 2014.
- [34] R. Kosara, F. Bendix, and H. Hauser. Parallel Sets: Interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006.
- [35] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale. Empirical studies in information visualization: Seven scenarios. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1520–1536, 2012.
- [36] J. Lawrance, C. Bogart, M. Burnett, R. Bellamy, K. Rector, and S. D. Fleming. How programmers debug, revisited: An information foraging theory perspective. *IEEE Transactions on Software Engineering*, 39(2):197–215, 2013.
- [37] B. Lee, N. Henry Riche, A. K. Karlson, and S. Carpendale. Spark-Clouds: Visualizing trends in tag clouds. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1182–1189, 2010.
- [38] A. Lewins and C. Silver. *Using software in qualitative research: A step-by-step guide*. SAGE Publications, 1st edition, 2007.
- [39] Z. Liu and J. Heer. The effect of interactive latency on exploratory visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2122–2131, 2014.
- [40] R. Lutz and S. Diehl. Using visual dataflow programming for interactive model comparison. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, pages 653–664, 2014.
- [41] P. Nguyen, K. Xi, A. Wheat, W. Wong, S. Attfield, and B. Fields. SensePath: Understanding the sensemaking process through analytic provenance. *IEEE Transactions on Visualization and Computer Graphics*, 22(01):41–50, 2016.
- [42] C. North. Toward measuring visualization insight. *IEEE Transactions on Computer Graphics and Applications*, 26(3):6–9, 2006.
- [43] M. Pohl and F. Scholz. How to investigate interaction with information visualisation: An overview of methodologies. In A. Ebert, C. G. van der Veer, G. Domik, D. N. Gershon, and I. Scheler, editors, *Building Bridges: HCI, Visualization, and Non-formal Modeling*, volume 8345, pages 17–29. Springer, 2014.
- [44] K. Reda, A. Johnson, J. Leigh, and M. Papke. Evaluating user behavior and strategy during visual exploration. In *Proceedings of the BELIV Workshop: Beyond Time and Errors – Novel Evaluation Methods for Visualization*, pages 70–77, 2014.
- [45] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, 2007.
- [46] M. Rosvall and C. T. Bergstrom. Mapping change in large networks. *PLoS one*, 5(1):e8694, 2010.
- [47] J. Saldana. *The Coding Manual for Qualitative Researchers*. SAGE Publications, 2nd edition, 2012.
- [48] M. Sandelowski, C. I. Voils, and G. Knaf. On quantizing. *Journal of Mixed Methods Research*, 3(3):208–222, 2009.
- [49] P. Saraiya, C. North, and K. Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):443–456, 2005.
- [50] M. Smuc, E. Mayr, T. Lammarsch, W. Aigner, S. Miksch, and J. Gartner. To score or not to score? Tripling insights for participatory design. *IEEE Transactions on Computer Graphics and Applications*, 29:29–38, 2009.
- [51] K. J. Srnka and S. Koeszegi. From words to numbers: how to transform qualitative data into meaningful quantitative results. *Schmalenbach Business Review*, 59, 2007.
- [52] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, 1981.
- [53] A. Telea and D. Auber. Code Flows: Visualizing structural evolution of source code. *Computer Graphics Forum*, 27(3):831–838, 2008.
- [54] J. J. Thomas and K. A. Cook. A visual analytics agenda. *Computer Graphics and Applications*, 26(1):10–13, 2006.
- [55] E. R. Tufté. *Beautiful Evidence*. Graphics Press, 1st edition, 2006.
- [56] C. Vehlou, F. Beck, P. Auwärter, and D. Weiskopf. Visualizing the evolution of communities in dynamic graphs. *Computer Graphics Forum*, 34(1):277–288, 2015.
- [57] N. Weibel, A. Fouse, C. Emmenegger, S. Kimmich, and E. Hutchins. Let’s look at the cockpit: Exploring mobile eye-tracking for observational research on the flight deck. In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pages 107–114, 2012.
- [58] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, 2003.