TU WIEN Informatics

Authoring Tool Prototype for Dashboard Onboarding Generation

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Medieninformatik und Visual Computing

eingereicht von

Ivaletta Shakhova
Matrikelnummer 01327973

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: DI Vaishali Dhanoa

Wien, 1. Mai 2023

Ivaletta Shakhova	Eduard Gröller

TU WIEN Informatics

Authoring tool prototype for dashboard onboarding generation

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Ivaletta Shakhova
Registration Number 01327973

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: DI Vaishali Dhanoa

Vienna, 1st May, 2023

Ivaletta Shakhova | Eduard Gröller

Erklärung zur Verfassung der Arbeit

Ivaletta Shakhova

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 1. Mai 2023

Ivaletta Shakhova

Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer, Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller, für die Unterstützung bei meiner Bachelorarbeit, das weitergegebene Wissens und die wertvollen Feedbacks herzlichst bedanken. Besonderer Dank gilt Vaishali Dhanoa und Vanessa Fediuk, die mir bei der Festlegung der Ziele und der Definition der Herausforderungen geholfen und mich in allen Phasen der Entwicklung begleitet und unterstützt haben.

Acknowledgements

Firstly, I would like to express my gratitude and appreciation to my supervisor, Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller, for conducting my bachelor's thesis, sharing the knowledge, and providing valuable feedback. Special acknowledgements go to Vaishali Dhanoa and Vanessa Fediuk, who assisted me in setting the goals and defining the challenges, guided and supported me in all phases of the development.

Kurzfassung

Datenvisualisierung ist ein effektiver Weg, um Einblicke in verschiedene Probleme zu gewinnen und den datengetriebenen Entscheidungsprozess zu fördern, allerdings kann diese Methode oft komplex und überfordernd sein. Um die korrekte Interpretation und das Verständnis von Daten zu ermöglichen, ist ein angemessener Onboarding-Prozess erforderlich. Onboarding schafft Klarheit über den Anwendungsbereich und den Zweck der Visualisierung für Benutzer mit unterschiedlichem Fachwissen. Allerdings kann die Erstellung eines Onboarding-Prozesses mithilfe eines Authoring-Tools eine Herausforderung darstellen, insbesondere für Autoren mit unzureichenden Fähigkeiten in Sachen Design.

In dieser Arbeit stellen wir ein Interface-Design für das Authoring-Tool im Rahmen des Dashboard-Onboarding-Prozesses vor. Wir haben die Herausforderungen analysiert, denen sich ein Neueinsteiger bei der Erstellung explorativer Erzählungen aus komplexen Daten gegenübersieht. Zu diesem Zweck haben wir bestimmte Storytelling-Szenarien aufgearbeitet, um die Aktivitäten nachzuvollziehen, die ein Autor an den Daten durchführen sollte, um zum Ziel zu gelangen. Wir konzentrieren uns auf die bedingte nicht-lineare Erzählung und wenden den Ansatz eines verzweigten Story-Graphen an, der als Knoteneditor implementiert ist. Die bereitgestellte Schnittstelle kann erweitert werden, um tiefgreifendere Storytelling-Strategien zu entwickeln und einen komplexeren und hochinteraktiven Onboarding-Prozess zu schaffen.

Abstract

Data visualization is an effective way to gain insights into various problems and improve the data-driven decision-making process, however it can often be complex and overwhelming. To enhance the interpretation and understanding of data, a proper onboarding process is required. Onboarding provides clarity on the visualization scope and purpose for users with different levels of expertise. The development of the onboarding process is facilitated with a suitable authoring tool. Though utilizing authoring tools, especially by creators with deficient visual design skills, can be quite challenging.
In this thesis, we introduce an interface design for the authoring tool within the scope of the dashboard onboarding process. We are focusing on the constrained non-linear narrative and applying the approach of a branching story graph that is implemented as a node editor. The provided interface can be extended to develop more profound storytelling strategies and aim to create a more complex and highly interactive onboarding process.

Contents

CHAPTER 1

Introduction

Data visualization is a powerful technique of data-driven problem-solving process and employs graphical elements such as plots, charts, graphs, etc. To display these graphical elements in an accessible format, a graphical user interface called dashboard tool is widely used [EAB13]. Dashboard tools, or simply dashboards, are used to analyze and overview data sets, and the process of explaining the data components to novices is called onboarding. The onboarding process can be carried out in many ways. One of the powerful techniques to carry out the onboarding process is storytelling, which as an effective approach for explaining disparate data is widely employed in various professional fields, especially for business purposes [BDF15]. Among the common onboarding techniques or means, can be presentations with explanation text or an author's voice-over, video presentations, or automated tours. Regardless of which approach is adopted, designing an onboarding process can be cumbersome for an analytical data expert who needs to prepare a thorough onboarding for various users.[Dha+22].

The primary objective of our authoring tool is to present the dashboard information in a way that can be easily adapted by the dashboard author to create engaging onboarding stories to fill the knowledge gap of the onboarded user.
Our contribution is to provide a congruous graphical user interface for the authoring stage of dashboard onboarding, that employs the process model proposed by Vaishali et al. [Dha+22]. Since the described onboarding technique is flexible, specific, and actionable [Dha+22], the introduced interface should assist this interactive storytelling approach [SMR03].

When using an authoring tool, usability and design are fundamental parts of interface development, since both have a great impact on user behavior and experience. Great authoring experience implies the production of impactful and comprehensible stories that are aligned with the author's objectives.
Thus, we have defined a set of requirements that our interface should meet:

- **Simplicity and Comprehensibility.** Since most users of the interface might not have much experience in creating visual narratives, the decision-making process should be reduced to the essentials. We need to avoid cluttered and overwhelming layouts and excessive information overload so that the interface communicates a sense of clarity [Gal07].
- **Unambiguity.** Each element of a successful design must serve a single purpose. Its meaning should be clear and unambiguous so that the user is not confused [Nie94]. Since we introduce icons to indicate major data components, tooltips have to be foreseen. The main purpose of tooltips is to convey the meaning of symbols through the user interface. To avoid any confusion about what a particular symbol represents, textual support is required. This considerably enhances the understanding of abstract symbols.
- **Intuitivity.** An interface is considered user-friendly if it reacts as expected. Users should not ponder for a long time what action they have to perform or element to click to get the information they need, but do so unconsciously. The user should be able to freely navigate and easily access the necessary components. Navigation bar tools should always be visible [Gal07].
- **Coherence.** Data components and interface elements must fit together in a coherent way, which means that particular elements belonging to a group should be recognised naturally [Nie94]. We use symbols and icons in our design that describe certain components and elements. We need to ensure that each element is always described by the same symbol throughout the interface.
- **Scalability and Flexibility.** The interface is supposed to be extensible [Gal07], which means that it is not only tailored to a specific scenario but can be adapted for various purposes. Parts of the interface must be encapsulated so that the interface possesses a high degree of integrity.

Pointing out and analyzing the challenges that a novice person faces when creating exploratory narratives of complex data is indispensable to developing a functional interface. For that purpose, we went through particular storytelling scenarios to understand the actions that an author should perform on the data to achieve their goals.

The first challenge we faced was information overload. All components and interactions must be visualized in such a way that the author can keep an overview without being overwhelmed. We also have to take into consideration that the stories can be arbitrarily extensive, thereby making the problem more complex. Graph-based systems are found to be ideal for such complex, interwoven narratives, as their ability to visualize data makes them the most accessible [GHC18].

The most challenging, however, is to allow an onboarded person to decide whether they want to have a closer look at specific data, skip some parts of the narrative, or review the narrative sequence. In this case, a deviation from the linear sequence should be

possible to achieve this level of interactivity. Unfortunately, we are not able to provide an interface for highly interactive dashboard onboardings at the current stage, therefore it is crucial to define the boundaries of end-user interactions.

Within the scope of our research, we want to achieve a set of goals: (1) fill the lack of existing authoring tools for producing data-driven explanatory narratives in the context of dashboard onboarding; (2) design and implement a user-friendly, efficient, clear, and expressive user interface for an automated onboarding solution proposed by Vaishali et al. [Dha+22], thus (3) supporting creators in developing appealing and comprehensible narratives that will be perceived by the end-user in the most efficient way; (4) allow analysts to create an adaptive and interactive dashboard onboarding process; and (5) meet the aforementioned design requirements and overcome the challenges.

CHAPTER 2

Related Work

The potential of exploratory narratives of visual data is vast and has not been thoroughly studied [Zha+22]. While reviewing the existing tools for generating narrative sequences in different areas, we found some ideas, approaches, and interface representations beneficial for our research.
We were interested in finding an authoring tool that could be combined with a process model for dashboard onboarding and that enables analytics to sequentially describe data visualizations such as charts, KPIs etc. by generating rule-based narratives and employing various presentation means.

Roslingifier [Shi+22] proposes "a data-driven narrative method for animated scatter plots" and introduces a semi-automated authoring tool to facilitate the production of powerful in-person presentations.

The Roslingifier interface, consists of three visual components: a Presentation Output View, an Event Exploration View, and a Presentation Editor, gives us a basic idea how an authoring tool for data presentations can look like.

The main components are called events, which consist of a group of entities and are depicted as gray rectangles. A story is a linear chronological presentation sequence of event segments. The events can, however, overlap or be displayed simultaneously. Users can add and delete events and change the order of the events or segments. These actions on data are essential when developing an authoring tool.

The authoring process begins with an automatically generated story, then the creator can edit and play the presentation interactively using *temporal branching*, *natural language narratives*, and *visual effects* methods.

The temporal branching technique delivers a certain level of interactivity, where the presenter sets rewinds at several time points in the narration sequence and adds other playback or highlighting techniques to describe other aspects of an event. This approach

seems to be practical for dashboard onboarding, however, it is not clear how it would perform if a great number of rewinds are utilized for multiple narrative trajectories. From our point of view, it could lead to a loss of overview.

The natural language narrative method is provided by the system in a written form. The presenter can add an explanation or a short description in the text field mapped directly into the presentation view. We found the design and placement of the description field reasonable, so we are going to incorporate it into our interface.

To highlight entities, trends, or insights *visual effects* such as labeling, spotlighting, tracing, and accumulation are used. Moving the position of labels and turning on and off the colored bubble labels is also possible. For now, we consider applying labeling and spotlighting to emphasize dashboard components and elements for different purposes.

From the very beginning of our research, we considered the presentation editor as part of the dashboard onboarding user interface, though in the mock-up phase we realized that this structure does not provide the desired level of interactivity of the dashboard onboarding process. The analysis of the Roslingifier user interface has revealed that the data-driven storytelling creator is not a perfect fit for the dashboard onboarding interface, but we can adapt some parts of the Roslingifier interface, such as the Presentation Output View, the narrative description field, and highlighting techniques.

Another authoring tool that has similarities with our approach is Novella proposed by Green [Gre18]. Novella is realized by using a node-based diagram, elements of a simple text editor, and directly mapped fields that require data entry. We found both the overall interface layout and most of the visual elements and interface features very conducive to the dashboard onboarding process model. Most of the authoring process is handled by the central node-based graph, with the nodes representing distinct forms of narrative elements. This kind of interface structure solves the problem of multiple narrative trajectories. The graph nodes are linked to each other through pins. Each node has at most one ingoing and at least one outgoing pin, which aids the development of interactive systems.

The nodes' data can be edited via popover windows, thus minimizing the need for screen space despite extensive content. If required, a popover can be detached to retain its visibility. We also foresee a field for a textual description of the dashboard component, so this feature might be useful.

The tool also provides undoing and redoing of the editing actions, which allows non-technical creators to learn the system by doing without being afraid of performing a wrong action.

Abstract data extracted from the system model that is mapped into nodes is indicated by a colored flag in the upper left corner. In this way, the type of node can be easily identified. We also strive for increased ease of data recognition, therefore we assign a distinct color and an icon to each dashboard component.

Another advantageous graphical feature that enhances the user experience is the highlighting of empty or incomplete states. No color is assigned to the unconnected pins, and unidentified nodes are labeled with a light placeholder text to indicate an incomplete state.

The Novella interface enables users to completely hide the panels that are employed less frequently, such as Story Preview, Variable Editor, and Entity Editor. This reduces the clutter of the user interface while working with complex data. We will definitely implement this feature in the development stage.

Nonetheless, we have found that no information about possible narrative means, such as human narration or screen recording and traversal strategies, was pointed out, which is an essential aspect of our research. Furthermore, the Novella interface offers superfluous features for the current stage of development of our interface, such as the trash function. Most content within the interface goes to the recycle bin before it is deleted. In our case, a deleted component moves to the component graph from the sidebar, since it must be used at least once in the narration sequence.

Despite the fact that Novella's interface meets most of our requirements and tackles some of the problems described, it cannot be leveraged for the dashboard onboarding process model without modifications. We would need to eliminate redundant functions, introduce dashboard components and interactions, change the appearance of the nodes, and implement some additional functions, such as the drag-and-drop feature.

Narvis Authoring Narrative Slideshows [Wan+18] enables teachers to produce narrative slideshows for introducing data visualization designs. Here, visual primitives, visual units, and visual views form a hierarchical structure of the model that reflects the component structure of the dashboard onboarding process model. The user interface consists of a Source Panel with input data, a unit panel, where the components' dependency is described, a Channel Panel where transitions and comments are added, and an Editing Panel that appears as the floating annotation window, where created annotations and transitions are modified.

The Narvis developers describe the relationships between the conceptual components, leading us to the notion of how we can achieve a certain degree of interactivity. They identify two types of relationships between visual units: independent, where no logic dependency exists between two visual units, and dependent, where one visual unit "A" depends on another visual unit "B" and must be explained before "B".

Since we are interested in the authoring phase, we analyzed the organizing and introducing units that are conducted in different panels.
The organization of the units is accomplished in the unit table, where the creators define the interrelationships between the units. The exact method of setting these relationships is, however, not further explained.

In introducing units, the Channels Panel and the Editing Panel are involved. The Channels Panel is responsible for displaying attributes of visual units that need to be

explained. To explain certain attributes, animated transitions, and annotations must be inserted and then manipulated in the editing area, i.e., resized or moved, for example. Even if the dashboard's components and interactions can be mapped out as visual channels, it is difficult to comprehend how a more extensive set of attributes explaining complex data could be displayed in such a compact format.

As we have learned from analyzing the Narvis authoring tools, defining relationships between visual elements can be helpful in producing interactive stories, but the complete tool and its interface are also not a perfect fit for a dashboard onboarding interface. Despite the extensive number of existing tools developed for generating visual narratives, we could not find any tool that fits the determined design requirements and addresses and tackles the described challenges within the scope of interactive onboardings.

CHAPTER 3

Mock-up Creation Stage

Throughout all the stages of the research process, diverse mock-ups were developed. For creating interface mock-ups, Adobe Illustrator [Ado23], a vector-based drawing software, and the graphic design platform Canva.com [PAO12] were employed.

The complete dashboard onboarding process involves two main user roles, namely, the author, who creates a narrative sequence, and the end-user, for whom a dashboard is designed and presented [Dha+22]. When developing an interface for the process model within this research, we only focus on the authoring part, which means no presentation preview design is foreseen.
At the initial stage of the research, we determined the primarily global interface structure, considering the Roslingfier [Shi+22] and Novella [Gre18] interfaces. The main layout is divided into three sections: Presentation Output View, Component Graph, and a Presentation Editor.

The main purpose of the Presentation Output View is to provide a preview of the generated narrative sequence. Furthermore, in this panel, the author can add annotations in a directly mapped text field shown in Figure 3.1. The text volume of the annotations needs to be restricted, as the presentation area offers limited preview space. Completely limiting the number of words is not something we want to pursue, since we want to provide the creators with more control over the authoring process. Instead, we put constraints in the form of a warning message (Figure 3.2).

The Component Graph consists of three levels of components [Dha+22] that build the hierarchy of the introduced onboarding process model: dashboard components, dashboard interactions, and dashboard data. As a rule, developers have to visualize extensive data on a relatively restricted layout. To save design space, the dashboard components and interactions are organized in collapsible tabs (Figure 3.3) and can be added to the narrative sequence by clicking on them or dragging and dropping them into the presentation editor.

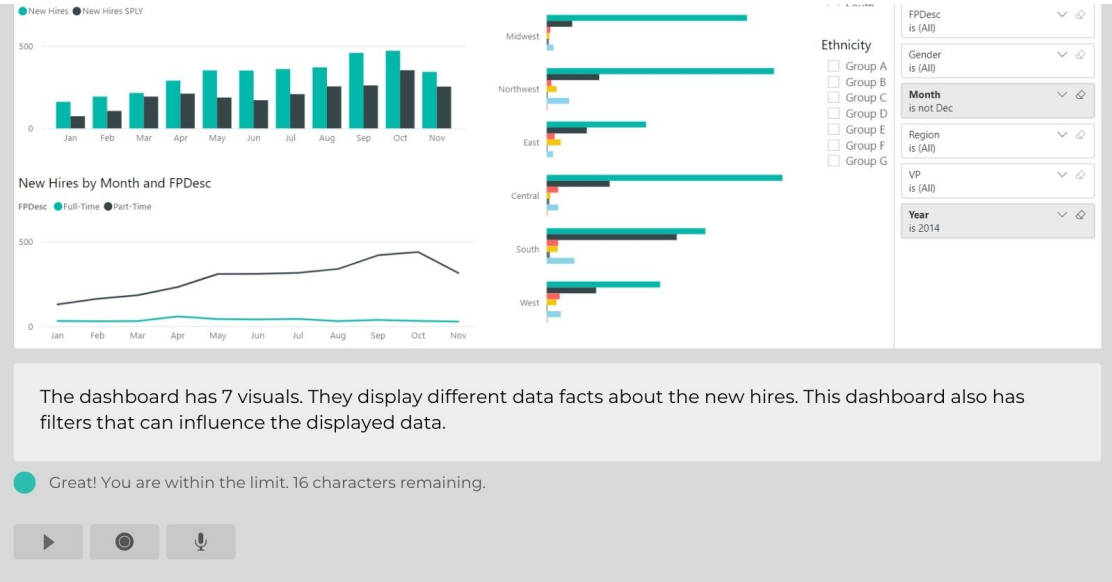

Figure 3.1: An annotation box, directly mapped into the Output View layout section.

Consequently, icons and symbols are employed for the sake of saving as much space as possible while still ensuring good readability (Figure 3.4). To improve the usability of our interface, we introduced a set of icons that identify data components. For the representation of the icons, metaphors from the real world are used to depict objects and activities in the digital world. We take advantage of previously learned correlations and reuse common or recognizable symbols so that users do not have to guess the meanings of functions. This ensures the coherence of the interface.

The design of the Presentation Editor was adopted from the Roslingfier interface layout. We have slightly modified its appearance so that it fulfills the functionality we have defined for onboarding purposes. The editor is depicted as a timeline with colored rectangles that represent different dashboard components (Figure 3.5). As the name implies, the entire authoring process takes place in this area.

As was already mentioned above, an element that describes a particular data component can be dragged and dropped from the component graph into the editor. By default, a group of elements describing the same main component is assigned a specific color. However, authors could assign their own color via a color picker, found in the component graph, to enhance the recognition of related elements. For instance, a bar chart is attributed a blue color. Consequently, the elements such as labels, axes, etc. that describe the bar chart are colored blue. All the elements of the Presentation Editor can be scaled, moved, or deleted. The scaling of the component determines the duration of its presentation in a narrative. To display several elements concurrently or time-shifted, the author could add several timeline tracks and place the rectangles parallel to each other on different tracks so that they overlap. This functional side of the Presentation

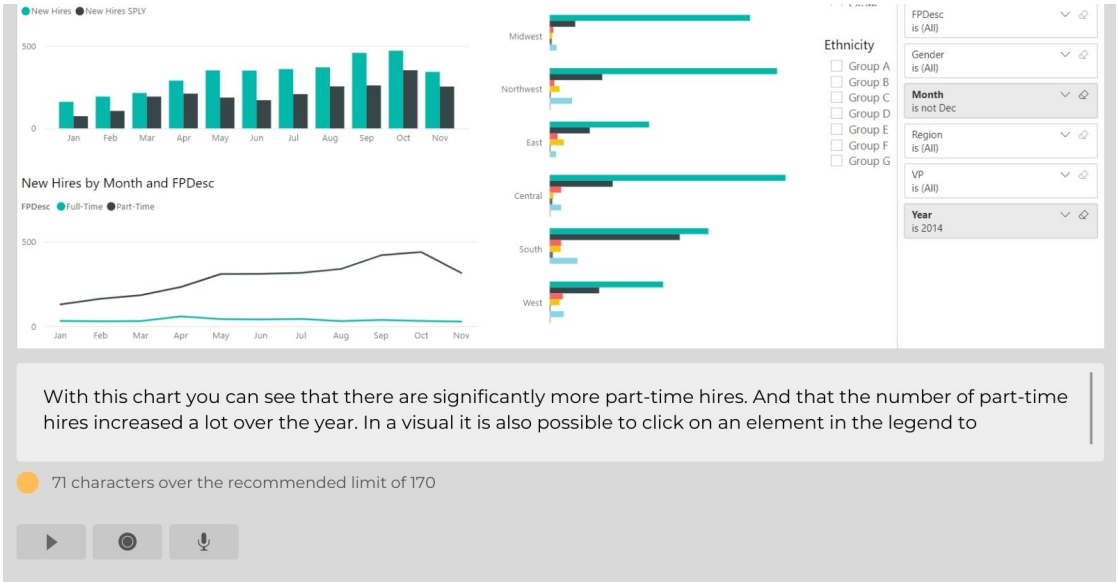

Figure 3.2: The warning message about exceeding the allowed length of the author's annotation.

Editor makes it suitable for producing narratives in linear, chronological order. However, after the cognitive walkthrough and delving deeper into the functions of the Presentation Editor, we faced a major challenge.

When an author conceives a story, the data arrangement is often intuitively determined, and the structure of the narrative is ordered according to the relevance of the data to certain end-user groups. Therefore, we should allow the author to stipulate certain conditions for narrative segments or to generate alternative narrative trajectories. It appeared to be troublesome to conceive of this scenario, and in our opinion, it is practically intractable to produce multiple storylines with the editor in timeline form. This problem prompted us to rethink the concept and redesign the interface.

After research, we found the solution – to develop an interactive narrative that allows the end user to steer the narrative flow [SMP03]. A suitable approach is to embody the data as a directed graph, where the nodes represent the dashboard components and interactions, and the arcs between the nodes represent the decisions that the end user can take. However, the amount of the narrative content can increase exponentially with the number of alternative trajectories, so that the authoring process becomes incomprehensible and infeasible [RB13]. Therefore, the dashboard onboarding system limits the degree of variation at certain parts of the narrative flow.

As we have already mentioned in the related work section, the Novella interface [Gre18] is implemented utilizing a node graph. Hence, we were looking for node editor designs that would serve our purposes and created a mock-up of the Node Editor whose design was adopted from the open source animation software Blender 3.0.1 [Com22] (Figure

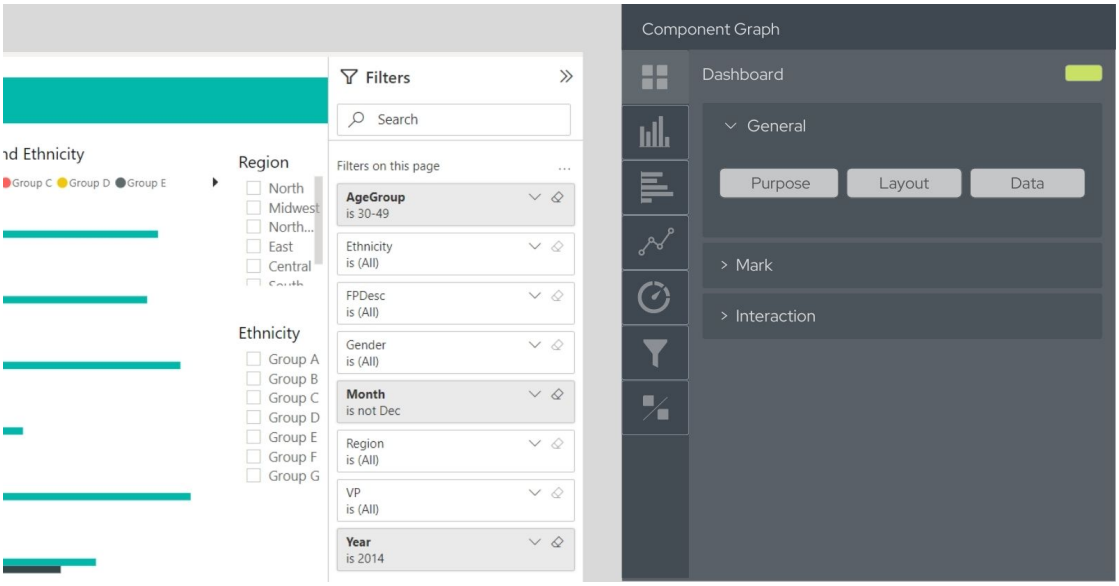

Figure 3.3: The Component Graph of the onboarding authoring tool with accordions for each dashboard component.

3.6). Each node in the Node Editor has a distinct color, heading, and icon that are used consistently throughout the interface so that it is recognized instantly on the canvas.

As the editor occupies the majority of the layout space, the Presentation Output View is moved to a separate, hidden window that can be accessed by changing the select option. In this way, the author can toggle between the Node Editor and the Presentation Output View.

By adopting the Node Editor, the need for the Presentation Editor is considered superfluous. However, we have not removed this panel, but have significantly reduced its functionality. The current purpose of the Presentation Editor or from this moment, renamed to a Narrative Sequencer, is to project an active narrative trajectory when the author clicks on a branch in the Node Editor. Additionally, the author is only able to set a duration of the specific elements in the narration flow by scaling them. To provide an indication of the currently active branch, the system highlights the element on which an action is being performed. For example, if the author clicks on an element in the narrative sequence, the corresponding node is highlighted in the Node Editor (Figure 3.7).

As we aim for semi-automatic authoring for dashboard onboarding, we need to specify an incomplete state of the entities after a primary narrative sequence has been created. Each dashboard component is required to be part of the narrative sequence. Consequently, if a component is missing, it can be highlighted in the component graph to indicate that the author has to manually insert it into the sequence (Figure 3.8). However, after some consideration, we have decided to omit this feature, as it makes the development process

Figure 3.4: Icons used throughout the interface that depict objects and activities in the digital world.

needlessly complicated at the current stage.

With this state of the design, the goal of enabling the author to generate rule-based narratives with variations so that the end user has the ability to decide how the narrative unfolds seems to be achievable. Thus, we decided to move on to the implementation phase.

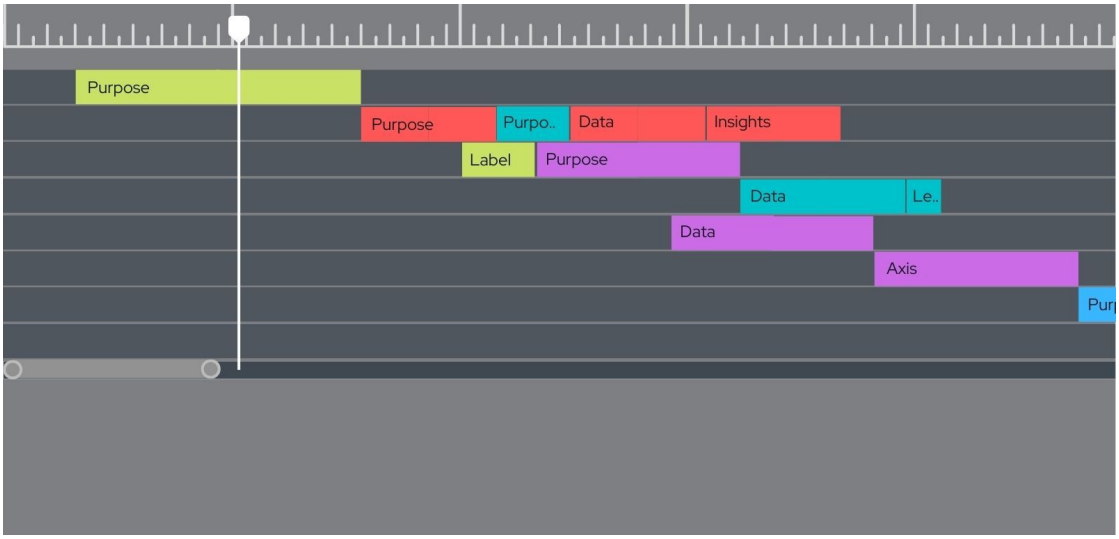

Figure 3.5: The Presentation Editor that was adopted from the Roslingfier [Shi+22] and modified according to authoring tool purposes.

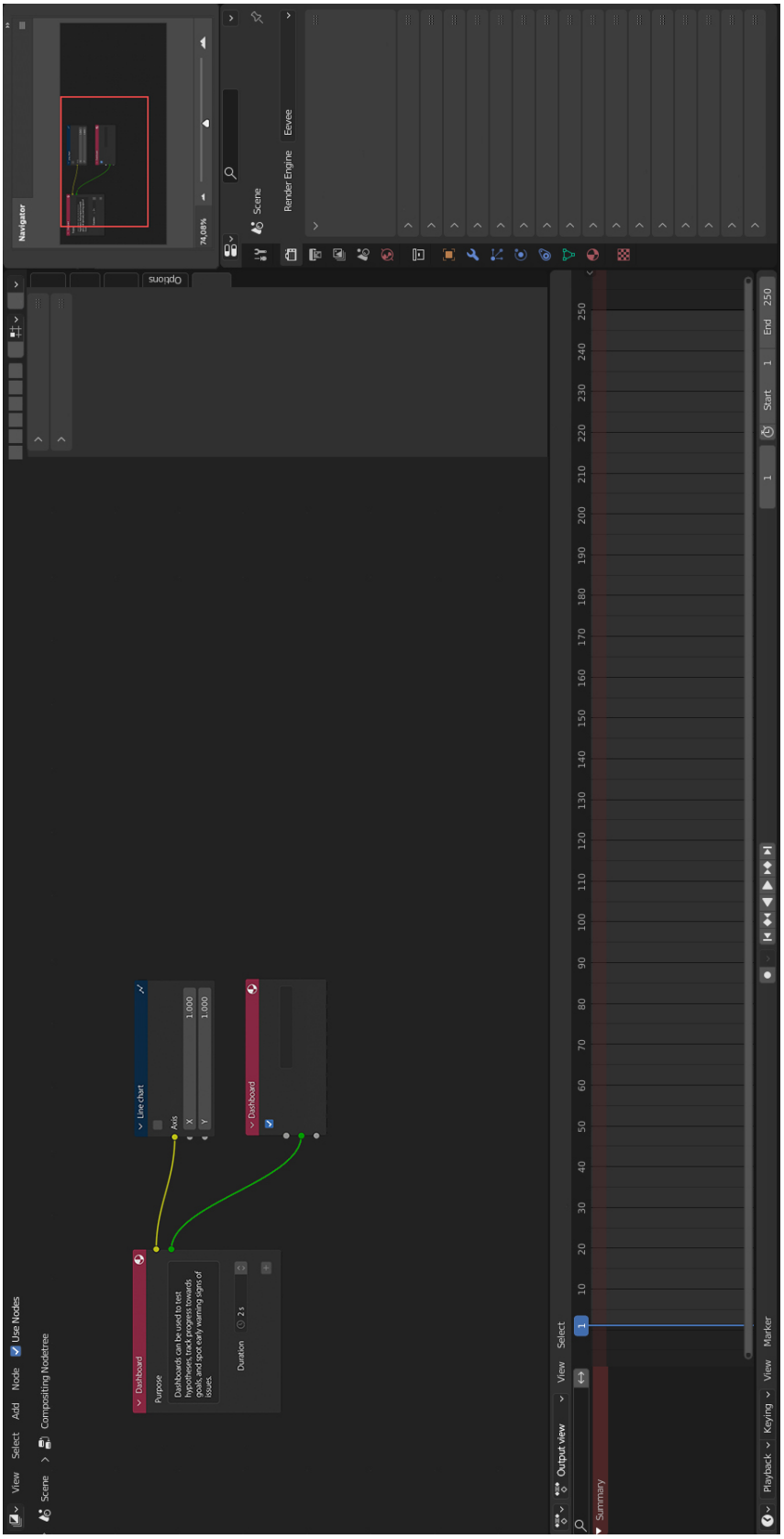

Figure 3.6: The Node Graph Editor that allows creators to produce different narrative trajectories.

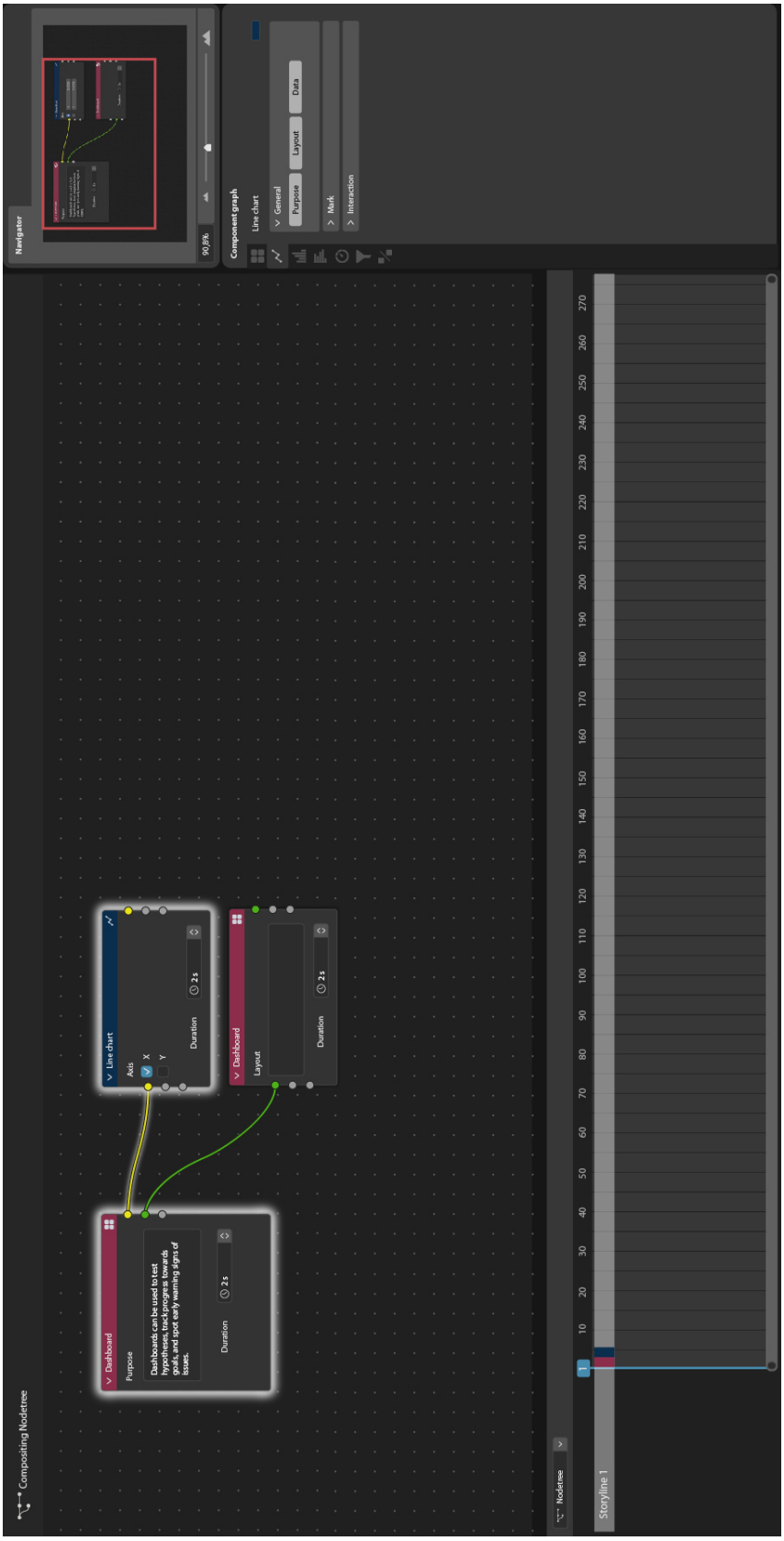

Figure 3.7: Node Graph Editor that allows creation to produce different narrative trajectories.

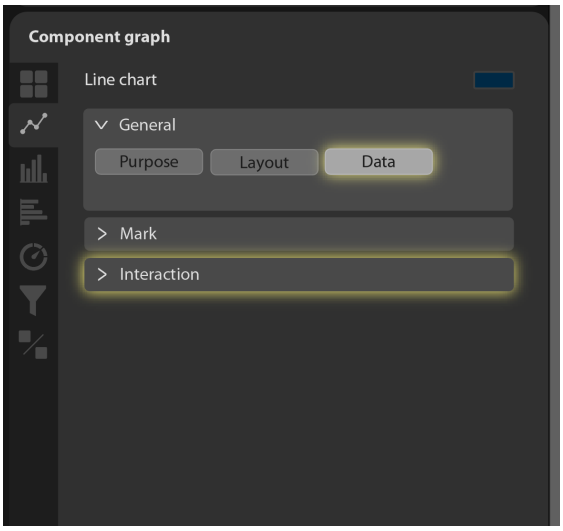

Figure 3.8: Incomplete state indication in the Component Graph after a narrative sequence is generated.

CHAPTER 4

Implementation and Results

In the development phase, we aim to incorporate the knowledge gained from the review of the related works, taking into consideration the described interface design and usability requirements, to tackle the identified challenges. To implement the graphical interface for the authoring tool, we utilise HTML, JavaScript, React, React Flow and Bootstrap libraries in the integrated development environment PHPStorm. The source code of the implemented part is available on GitHub [Dha23].

The first step towards deployment of the authoring tools is to set up a basic layout with three panels: the left panel for displaying the traversal strategies and entities of the Component Graph (Figure 4.1a), the middle panel for the Node Editor (Figure 4.1b), and the right panel for the Presentation Output View (Figure 4.1c). To achieve that, we create React components for resizable panels, set the initial width and also minimum and maximum width values to avoid content overlap.

```
1 <ResizePanel initialWidth={250} maxWidth={400} minWidth={250}>
2 <ResizeContent/>
3 </ResizePanel>
```

The left panel is organized into several sections for the traversal strategy options and for the dashboard components and sub-components. As there are many components to be displayed in the component diagram, we need to spare space, therefore we pack the elements into collapsible Bootstrap accordions.

The Node Editor is the main part of our system, where most of the authoring process is accomplished. It is implemented with the aid of an interactive, node-based editor provided by the customizable library React Flow [web22]. The node canvas on which all node sequences are rendered is generated by adding a `<ReactFlow/>` component.

React Flow offers different built-in node types, depending on the number of handles [web22]. We decide to remove the incoming and outgoing handles and therefore implement

Figure 4.1: First prototype of the Dashboard Onboarding Authoring Tool consists of three resizable panels: the Component Graph, the Node Editor, and the Output Editor.

a custom node, which is additionally assigned a title and a corresponding color. Since we want to achieve a certain degree of interactivity, the system should allow the setting of change points and relationships between node segments.

Another type of node that we use is a group node. Since we want to provide a certain degree of interactivity, the system should allow the setting of change points and relationships between the nodes or the node segments. To achieve this, nodes are grouped together and a desired type of connectivity is specified (Figure 4.2). The segments that define different alternative narrative trajectories are arranged parallel to each other. One of three types of connectivity can be assigned to a group of nodes. The author can define whether *all*, *at least one*, or *only one* sequence of the group must be traversed in arbitrary order.

To render the nodes on the canvas, we pass base properties with the initial nodes to the `<ReactFlow/>` component, and since the native nodes have no default styling, we add CSS classes and attributes to render the nodes as they appear in mock-ups:

```
1  <ReactFlow
2   nodes={nodes}
3   onNodesChange={onNodesChange}
4   nodeTypes={nodeTypes}
5   onInit={setReactFlowInstance}
6   snapToGrid
7   fitView
8   >
9   </ReactFlow>
```

React Flow contains a variety of built-in functions that we utilize and extend. All

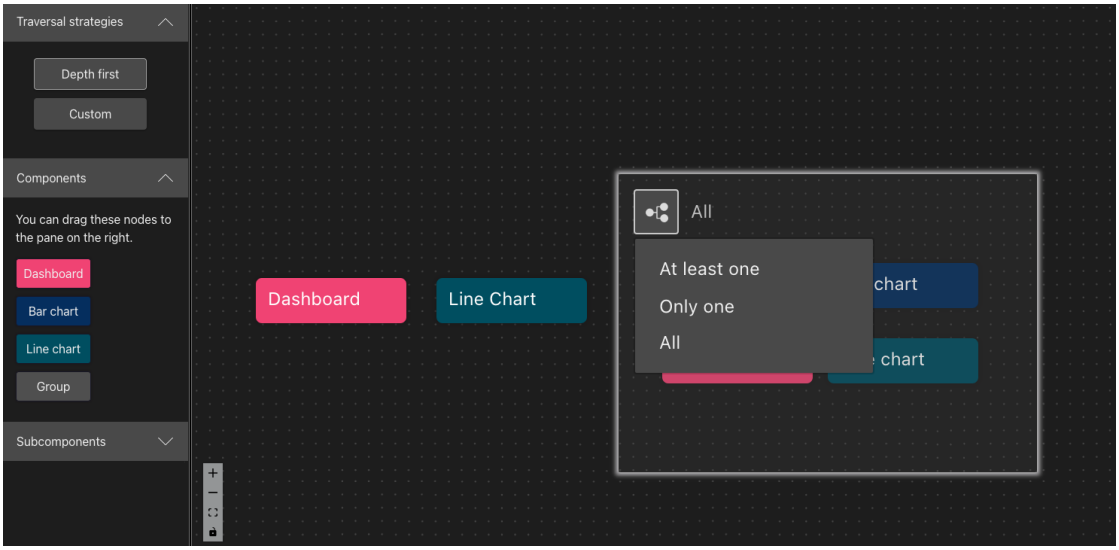

Figure 4.2: The Dashboard Components can be grouped together to define a particular dependency between them.

methods are called by passing the properties to the <ReactFlow/> component. Moving nodes around the canvas and selecting and deleting nodes are already supported by the library. However, to delete a node, the creator should press the backspace key, which seems intuitive, although not sufficient from our point of view. Therefore, we create a context menu that appears on right-click on a node and enables the author to delete any node, including group nodes. The action is invoked by passing the property onNodeContextMenu. Afterwards, we append a delete option to the context menu. To delete a node from the canvas, we simply update the node array by filtering all nodes with a different ID than the node to be removed.

```
1  const [position, setPosition] = useState({x:0, y:0});
2  const [isOpen, setIsOpen] = useState(false);
3
4  <ContextMenu
5  isOpen={isOpen}
6  position={position}
7  onMouseLeave={()=>setIsOpen(false)}
8  actions={[{label: 'Delete', effect:deleteNode}]}>
9  </ContextMenu>
```

The significant function that we want to provide is dragging the nodes from the Component Graph into the Canvas. The drag-and-drop option outside of the React Flow area is not integrated, although it can be implemented with the native HTML Drag and Drop API [web22].

First, we determine which elements should be draggable by assigning the draggable attribute and the onDragStart event handler to the nodes in the Component Graph.

With the `onDtagStart` event handler, we pass the event's data, i.e., the type and the title of the node, to create a corresponding node on the canvas after dropping:

```
<div className="dndnode dashboard"
onDragStart={(event) => onDragStart(event, 'simple', 'dashboard', '
    Dashboard')}
draggable>
Dashboard
</div>
```

The dropping behavior is realized with the `onDrop` event handler, where the node data is retrieved and processed. Then a new node is created and added to the array of nodes accordingly.

```
 const onDrop = useCallback(
    (event) => {
        event.preventDefault();

        const reactFlowBounds =
            reactFlowWrapper.current.getBoundingClientRect();
        const type = event.dataTransfer.getData('nodeType');
        const data = event.dataTransfer.getData('data');
        const title = event.dataTransfer.getData('title');

        // check if the dropped element is valid
        if (typeof type === 'undefined' || !type) {
            return;
        }

        const position = reactFlowInstance.project({
            x: event.clientX - reactFlowBounds.left,
            y: event.clientY - reactFlowBounds.top,
        });
        const newNode = {
            id: getId(),
            type,
            position,
            data: {
                title: title,
                type: data,
            },
        };

        setNodes((nds) => nds.concat(newNode));
    },
    [reactFlowInstance]
);
```

The React Flow library contains many useful features that we find very helpful, such as controls for the Node Editor like zooming and panning. The control panel is always accessible to the author, which helps to maintain an overview. For even better orientation across the graph, a mini map can be integrated. The minimap is a React Flow component

which provides an interactive minimap to React Flow that displays the entire flowchart [web22].

The implementation of the preliminary designs involves constant changes, e.g., the annotation field is moved from the Output View to the foot of the Node Editor. The annotation box appears as soon as a node is selected, which saves space and reduces UI clutter.

In the first stage of prototype development, we achieved to implement the basic functionality of the authoring tool. With the current state of the user interface it is possible (1) to display the dashboard components in the Component Graph as nodes with predefined attributes: the title and corresponding color; (2) to drag the nodes from the Component Graph and drop them into the canvas in the Node Editor panel ; (3) to select and deselect the nodes, move them freely around the canvas and delete them; (4) to group the nodes and define the desirable dependency between them by choosing the corresponding option in the context menu; (5) to pan the Node Editor canvas by dragging and zoom in or out by using React Flow controls.

Since this research is an ongoing process, the interface design and its functionality will be drastically modified and expanded in the further stages of development to facilitate an understanding of the analytical data.

CHAPTER 5

Conclusion and Future Work

In this thesis, we have created a first prototype of the authoring tool interface for generating dashboard onboardings. We have learned what challenges non-technical creators face while producing visual explanations of complex analytical data and how they can be overcome by using a well-designed interface.

We intend to continue extending the authoring tool functionality and improving the usability of the interface design. Learning from the observations made in this paper, some improvements are already planned. We will implement state notification to facilitate the exploration of constructed narratives and encourage learning from experience with the system. The state notification includes undoing and redoing state changes or displaying an alert message if an action is not reversible. Then the presentation means, such as guided tours, audio and video presentations, and scripts and tree traversal strategies, will be added.

As system functionality increases, so do the demands on design, therefore, usability has to be constantly refined to ensure a great user experience.

List of Figures

Bibliography

[Ado23] Adobe Inc. *Adobe Illustrator*. Version CC 2020 (24.0.2). Mar. 25, 2023. URL: https://adobe.com/products/illustrator.

[BDF15] Jeremy Boy, Francoise Detienne, and Jean-Daniel Fekete. "Storytelling in information visualizations: Does it engage users to explore data?" In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pp. 1449–1458.

[Com22] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2022. URL: http://www.blender.org.

[Dha+22] Vaishali Dhanoa et al. "A Process Model for Dashboard Onboarding". In: *Computer Graphics Forum (EuroVis '22)* 41.3 (July 2022), pp. 501–513. DOI: 10.1111/cgf.14558. URL: https://doi.org/10.1111/cgf.14558.

[Dha23] Vaishali Dhanoa. *Dashboard Onboarding UI*. 2023. URL: https://github.com/jku-vds-lab/dashboard-onboarding-ui (visited on 05/01/2023).

[EAB13] Micheline Elias, Marie-Aude Aufaure, and Anastasia Bezerianos. "Storytelling in visual analytics tools for business intelligence". In: *Human-Computer Interaction–INTERACT 2013: 14th IFIP TC 13 International Conference, Cape Town, South Africa, September 2-6, 2013, Proceedings, Part III 14*. Springer. 2013, pp. 280–297.

[Gal07] Wilbert O Galitz. *The essential guide to user interface design: an introduction to GUI design principles and techniques*. John Wiley & Sons, 2007.

[GHC18] Daniel Green, Charlie Hargood, and Fred Charles. "Contemporary issues in interactive storytelling authoring systems". In: *Interactive Storytelling: 11th International Conference on Interactive Digital Storytelling, ICIDS 2018, Dublin, Ireland, December 5–8, 2018, Proceedings 11*. Springer. 2018, pp. 501–513.

[Gre18] Daniel Green. "Novella: an authoring tool for interactive storytelling in games". In: *Interactive Storytelling: 11th International Conference on Interactive Digital Storytelling, ICIDS 2018, Dublin, Ireland, December 5–8, 2018, Proceedings 11*. Springer. 2018, pp. 556–559.

[Nie94] Jakob Nielsen. “Enhancing the explanatory power of usability heuristics”. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 1994, pp. 152–158.

[PAO12] Melanie Perkins, Cameron Adams, and Cliff Obrecht. 2012. URL: https://www.canva.com/.

[RB13] Mark Owen Riedl and Vadim Bulitko. “Interactive narrative: An intelligent systems approach”. In: *Ai Magazine* 34.1 (2013), pp. 67–67.

[Shi+22] Minjeong Shin et al. “Roslingifier: Semi-Automated Storytelling for Animated Scatterplots”. In: *IEEE Transactions on Visualization and Computer Graphics* (2022).

[SMP03] Daniel Sobral, Isabel Machado, and Ana Paiva. “Managing authorship in plot conduction”. In: *Virtual Storytelling. Using Virtual RealityTechnologies for Storytelling: Second International Conference, ICVS 2003, Toulouse, France, November 20-21, 2003. Proceedings 2*. Springer. 2003, pp. 57–64.

[SMR03] Nicolas Szilas, Olivier Marty, and Jean-Hugues Réty. “Authoring highly generative interactive drama”. In: *Virtual Storytelling. Using Virtual RealityTechnologies for Storytelling: Second International Conference, ICVS 2003, Toulouse, France, November 20-21, 2003. Proceedings 2*. Springer. 2003, pp. 37–46.

[Wan+18] Qianwen Wang et al. “Narvis: Authoring narrative slideshows for introducing data visualization designs”. In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 779–788.

[web22] webkid. *React Flow, a library for building interactive node-based UIs*. webkid GmbH. Kohlfurter Straße 41/43, 10999 Berlin, 2022. URL: https://reactflow.dev/ (visited on 05/01/2023).

[Zha+22] Yangjinbo Zhang et al. “A Visual Data Storytelling Framework”. In: *Informatics*. Vol. 9. 4. MDPI. 2022, p. 73.