TU WIEN Informatics

Interaktives 3D Storytelling für die Erforschung von Planetenoberflächen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Martin Mautner, Bsc
Matrikelnummer 1127229

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Eduard Gröller, Ao.Univ.Prof. Dipl.-Ing. Dr.techn.
Mitwirkung: Thomas Ortner, Msc Msc

Wien, 4. Mai 2020

Martin Mautner	Eduard Gröller

TU WIEN Informatics

Interactive 3D Storytelling for Planetary Exploration

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Martin Mautner, Bsc
Registration Number 1127229

to the Faculty of Informatics

at the TU Wien

Advisor: Eduard Gröller, Ao.Univ.Prof. Dipl.-Ing. Dr.techn.
Assistance: Thomas Ortner, Msc Msc

Vienna, 4th May, 2020

Martin Mautner	Eduard Gröller

Erklärung zur Verfassung der Arbeit

Martin Mautner, Bsc

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, 4. Mai 2020

Martin Mautner

Acknowledgements

I would like to thank everyone who supported me with this work, both with technical expertise and moral encouragement. Special thanks goes to Thomas Ortner who always seemed to have the right answers when issues related to visualization design popped up, and showed incredible patience with my slow writing speed. I would also like to thank the Aardvark development team for helping with technical issues, in particular Harald Steinlechner for listening to and taking care of my convoluted bug reports. Andreas Walch was of immense help towards the end by reading this thesis multiple times, pointing out grammar and spelling mistakes, as well as suggesting better formulations for unclear parts. Last but not least, I would like to thank my parents for making this work and my whole course of studies possible.

Kurzfassung

Der Planetary Robotics 3D Viewer (PRo3D) ist ein interaktives Tool zum Durchführen von geologischen Analysen an Planetenoberflächen. Das Hauptziel ist es, Geologen und Geologinnen der NASA und ESA dabei zu unterstützen, Leben auf dem Mars nachzuweisen, indem Analysen an einem hochauflösendem 3D-Modell der Marsoberfläche durchgeführt werden können. PRo3D erleichtert explorative Analysen von großen Datensätzen, jedoch gibt es keine Funktionalitäten, die das Kommunizieren von neuen Entdeckungen ermöglicht. In dieser Diplomarbeit erörtern wir den Entwurf und die Implementierung von Storytelling-Mechanismen in PRo3D, welche eine einfache, schnelle und interaktive Kommunikation von Ergebnissen erlauben. Außerdem zeigen wir wie der Analyseprozess selbst, der zu den Ergebnissen führte, mittels Provenance in geologische Stories eingearbeitet werden kann. Dadurch können die einzelnen Schritte der Analyse erläutert und begründet werden, wodurch die Verifizierbarkeit und Reproduzierbarkeit der Ergebnisse erleichtert wird. Wir geben einen Überblick über Storytelling und Provenance im Kontext von Visualisierung. Weiters untersuchen wir verschiedene Ansätze beim Entwurf von Provenance-basiertem Storytelling im Rahmen von geologischen Untersuchungen, welche in PRo3D durchgeführt werden. Schließlich präsentieren wir unseren Prototypen, der auf diese Überlegungen aufbaut.

Abstract

The Planetary Robotics 3D Viewer (PRo3D) is an interactive visualization tool that allows for geological analyses of planetary surfaces. The primary goal is to support geologists at NASA and ESA in their mission to find signs of life on Mars by enabling them to perform analyses on a high-resolution 3D surface model. While PRo3D facilitates an exploratory workflow to gain new insights, there is a lack of support to communicate new findings. In this thesis, we discuss the design and implementation of storytelling mechanisms into PRo3D that allow for an easy, fast, and interactive communication of results. Moreover, we show how provenance information can be incorporated into stories, enabling geoscientists to present how they arrived at a certain discovery or interpretation. Provenance includes the individual steps in the analysis process that lead to a given finding, supporting its verification and reproducibility. We present a broad overview about storytelling and provenance in visualization, and discuss the design space of a provenance-based storytelling approach in the context of geological analyses as conducted in PRo3D. Finally, we present a prototype as a proof of concept based on these deliberations.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Sedimentary structures discovered during the Mars Exploration Rover (MER) mission by the National Aeronautics and Space Administration (NASA) share similarities with formations found on Earth [GAB+05]. This motivates the application of the methods and principles used in sedimentology and stratigraphy to gain insights about the geological history of Mars [HGE+11]. Finding evidence for environments with rich and lasting water resources is of particular interest, as water is a prerequisite for life as we know it [GSK+14]. Geologists have to resort to analyses based on imagery obtained by the rovers during the MER and Mars Science Laboratory (MSL) missions, since field studies cannot be conducted on Mars. Outcrops, visible exposures of bedrock above the surface, are therefore studied to directly observe the sedimentary structures of an area. Outcrop analysis includes annotating and measuring properties such as geometry, distribution, and scale of these structures [HGE+11].

The Planetary Robotics Vision Data Exploitation (PRoViDE) project [PMT+15] aims to facilitate this by compiling the available imagery and reconstructing digital 3D models of the Martian surface. As the reconstructions combine large-scale orbiter imagery with high-resolution rover imagery, geoscientists may analyze outcrops at millimeter to centimeter scale while retaining large spatial context at the same time [PMT+15, BGT+18]. The surface models are saved in the Ordered Point Cloud (OPC) format, which was designed to represent data from multiple sources in multiple resolutions [BGT+18]. Figure 1.1 shows two examples of digital surface models that were reconstructed by combining orbiter and rover imagery using PRoViDE.

The Planetary Robotics 3D Viewer (PRo3D) [BGT+18] is another essential component of PRoViDE, which allows for the visualization of the reconstructed OPC models. Level of detail (LoD) rendering makes it possible to explore large datasets at interactive frame

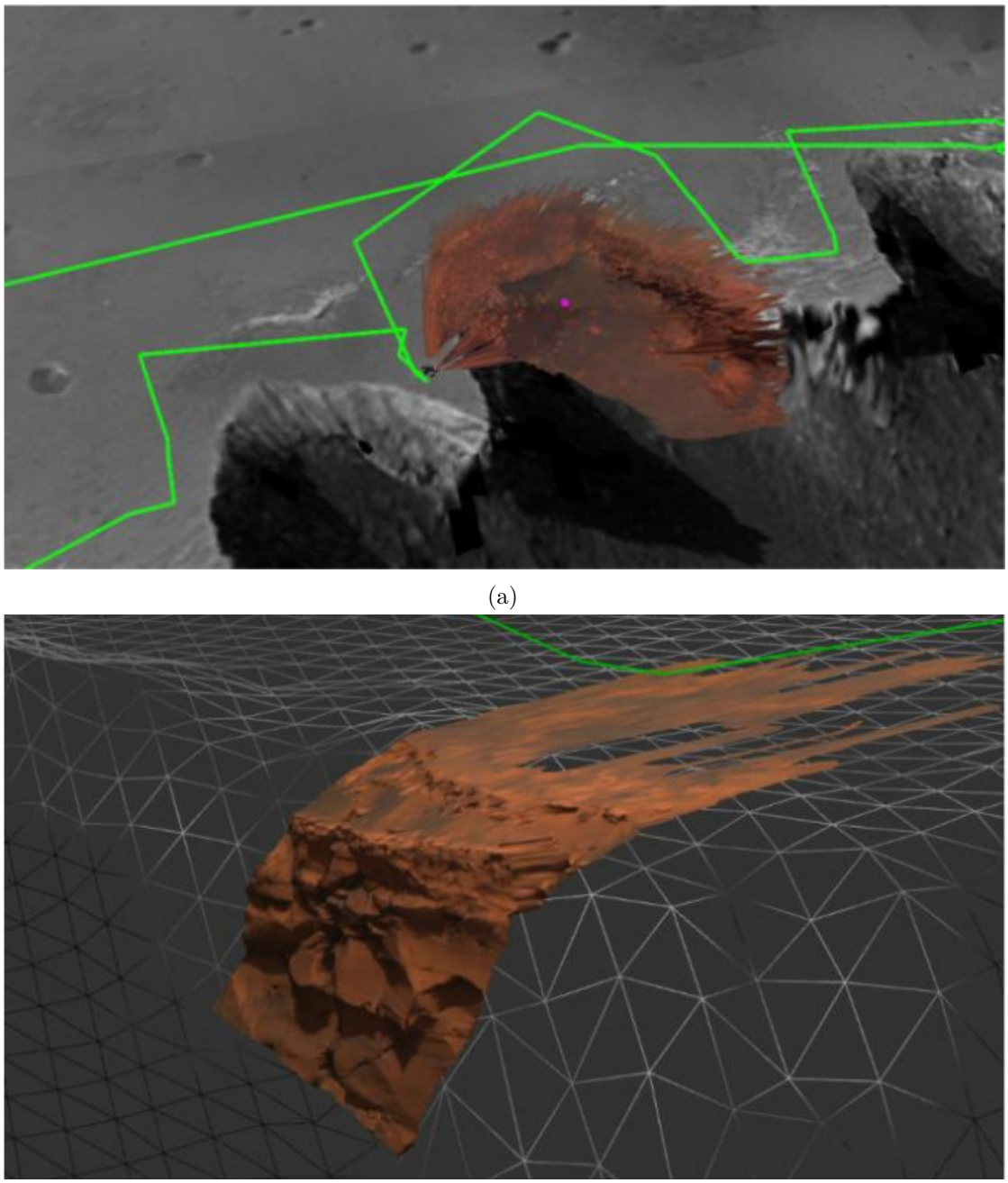


Figure 1.1: Surface models reconstructed by combining (a) High Resolution Imaging Science Experiment (HiRISE) and MER imagery, (b) HiRISE Digital Terrain Model (DTM) and MER wide-baseline stereo imagery.

Source: Paar et al. [PMT+15]

rates. In addition to viewing the 3D models, PRo3D allows users to measure typical quantities used in outcrop analysis. Using the currently supported measurement tools, a user may:

- measure the length of a line between two points on the surface. It is also possible to display the vertical distance and slope of that line.
- project polylines onto the surface, either (1) vertically, or (2) from the viewpoint of the camera. Subsequently, the length of the resulting polyline is returned.
- determine the orientation of a planar surface (e.g. a bedding or fault plane) by measuring its dip and strike values. These measurements describe the angle of inclination and the compass direction of a surface (i.e. the orientation of the line representing its intersection with a horizontal plane) respectively.

Furthermore, annotations can be placed directly on the surface. This includes polylines and glyphs from the measuring tools, but also simple text labels. The weight and color of the lines can be adjusted to indicate different types of boundaries. These features are exemplified in Figure 1.2.

1.2 Problem statement

The features provided by PRo3D facilitate an exploratory workflow to gain new insights. However, there is a lack of support to communicate new findings. This may be required if analysts want to explain their thought process to a colleague or present discoveries to a general audience. Such presentations usually make use of slide shows that are created using separate tools such as Microsoft's PowerPoint. In order to show data from analyses performed within PRo3D, the geologist has to capture screenshots or videos, edit them using image and video editing software, and finally embed them into the slide show. This workflow is suboptimal, primarily due to the following problems:

- Editing artifacts such as images and videos is a time-consuming and tedious task. This is especially the case for videos, as video editing requires knowledge and skills that are unlikely to be in the repertoire of the average user.
- Both screenshots and videos are static in what data they show. The author of the presentation has to decide a priori what aspect of the data is relevant. This is problematic if the prepared data are not sufficient to answer questions from the audience.
- Screenshots cannot capture the whole spatial context of the 3D data on their own. This is illustrated by the two screenshots in Figure 1.3, which were taken in PRo3D and used in a slide show presentation. First, an overview of an outcrop is shown in Figure 1.3a. It is divided into separate units denoted by the pink lines. Figure 1.3b

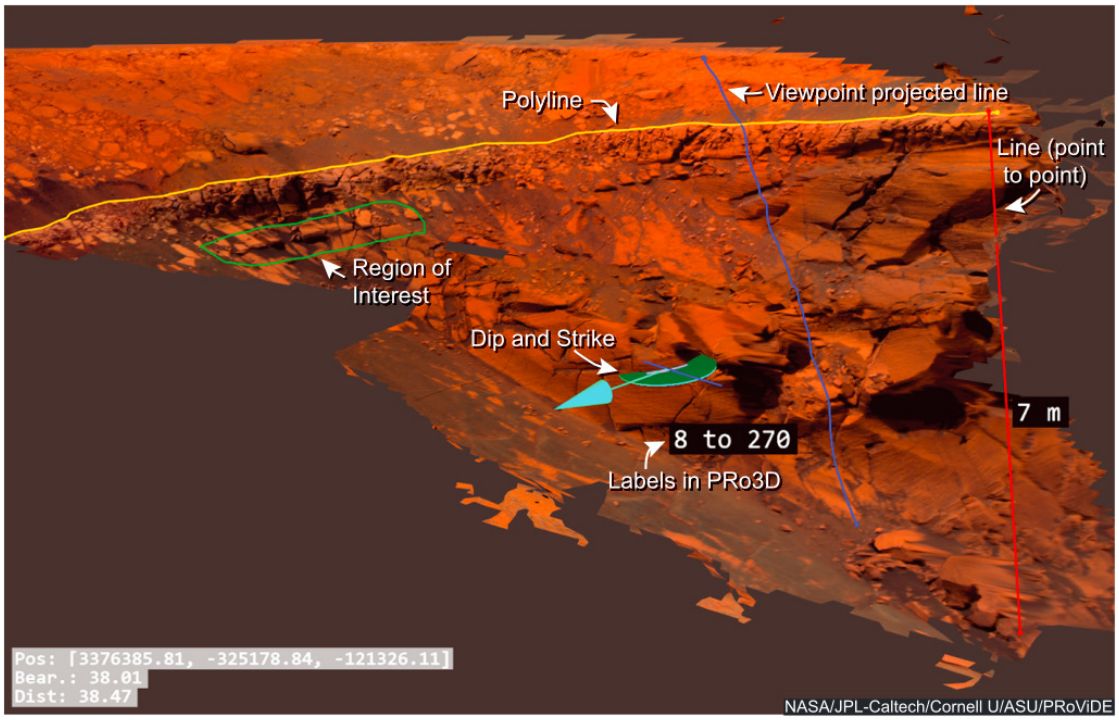


Figure 1.2: Overview of annotation features present in PRo3D. Note that the white labels and arrows were added separately and are not part of PRo3D.

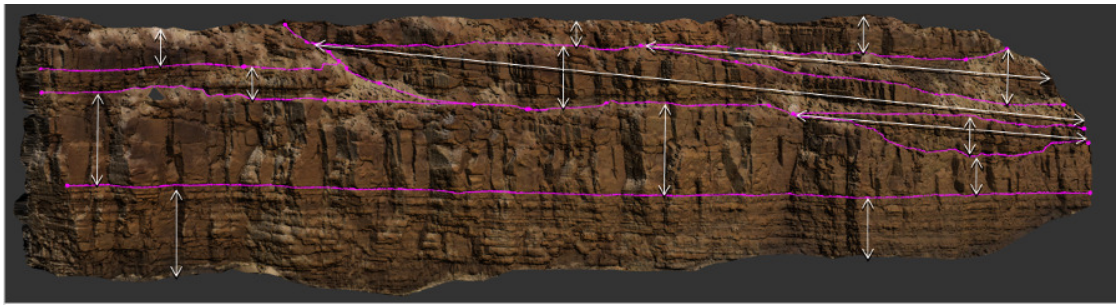
Source: Barnes et al. [BGT+18]

shows a zoomed-in view of one of the units. It is not immediately clear which unit is shown in the second screenshot. In order to preserve the spatial context between the two screenshots, additional annotations are required.

A possible solution to these problems is to integrate *storytelling* or *narrative visualization* mechanisms directly into PRo3D. Instead of using visualization solely as a means for exploration, data can also be visualized to tell a story. In the case of PRo3D, interactive, annotated animations can be used to communicate the findings and thought process of geologists as a visual story. To achieve this, PRo3D has to be extended to allow users to create such stories based on the conducted analysis.

1.3 Aim of the work

In this thesis, we discuss the design and implementation of storytelling mechanisms into PRo3D that allow for an easy, fast, and interactive communication of results. Our solution lets users build geoscientific stories as slide shows directly within PRo3D. Individual slides show the outcrop dataset from a specific point of view augmented with annotations



(a) Overview of the outcrop.

(b) Detail view of one of the units

Figure 1.3: Two screenshots taken from PRo3D as used in a slide show presentation. First an overview of the whole outcrop is shown in (a). In the next slide, a zoomed-in view of one the units is shown in (b). The spatial context between both views is lost by using static screenshots.

Source: Presentation by Barnes 2015

that the geologists added during their analysis of the data. Interactive controls and animations between the slides make it possible to present the visual data in a flexible way and preserve their spatial relations. Additionally, our solution provides means that let geoscientists communicate the mental process that lead to a given result. To this end, we combine provenance information of the analysis session with visual data stories similar to the Capture, Label, Understand, Explain (CLUE) model [GLG+16]. Provenance is the lineage of a datum [SPG05]. It describes every source and intermediate step of an element's derivation history, and which other data are derived from it. Provenance of an analysis session in PRo3D encompasses every stage in the interpretation workflow that resulted in the final geological model. This can be represented as a graph with branches denoting multiple alternative interpretation possibilities. The user can create a geological story by selecting an arbitrary linear path through the provenance graph and adding descriptive annotations to each slide. This associates any part of the story with a specific analysis state, which is crucial for a scientific workflow. The scientific method is iterative, meaning that discoveries are used as a foundation for further research to gain new insights [DC02]. A central aspect of this workflow is verifying the correctness of results by reproducing them [Ple18]. Our integrated analysis and provenance-based storytelling solution facilitates this, as a consumer of a story may return to the analysis stage from any point of that story. Subsequently, they may verify the presented facts or come up with an alternative geological model by building upon the story.

1.4 Structure of the work

In Chapter 2, we describe the concept of storytelling in the context of information and scientific visualization. We present a concise definition of visual data stories and explore the workflow and scenarios of narrative visualization in more detail. We also discuss how storytelling can integrate user interaction, and finally present various examples of visual storytelling in practice. Chapter 3 addresses provenance; we give an overview of provenance, and present a taxonomy according to the type of data that is captured and how the information is used. A crucial aspect of provenance is how valuable portions of the data can be filtered and queried. We examine different approaches for querying provenance and present practical examples thereof. In Chapter 4, we explore the design space of provenance-based storytelling for geoscientific analyses. We propose five key requirements for a storytelling solution in PRo3D, based on the geoscientific analysis workflow, already existing presentations, and domain expert feedback. The chapter also covers individual design points in more detail: how a visual story is structured in PRo3D, how provenance is captured and queried, and the role of the 3D camera. Chapter 5 presents our prototype based on the deliberations of the previous chapters. We reason about our decisions concerning the visualization design, explain technical implementation details, and finally present a geological story created with our prototype. Lastly, we give closing thoughts, mention limitations and possible future work in Chapter 6.

CHAPTER 2

Storytelling

Traditionally, visualization has been used to gain insights from data that are too large or too complex to analyze directly [KM13, TC06]. Visualization enables an exploratory workflow; that is, data scientists may start their analysis without initially knowing what they are looking for in particular. The data can be viewed from different points of views and with varying parameters to search for interesting features that are otherwise not apparent. This workflow has been the topic of research within the information and scientific visualization communities for the past few decades. At this point, the design space of data visualization for exploration is well understood [KM13]. Accordingly, there are suitable visualization methods for most types of data and tasks [KM13]. While visual exploration facilitates making discoveries, these discoveries also have to be shared with others to have an impact in a scientific workflow. The communication of findings can happen in various forms and settings. For example, scientists might want to share their analysis results with colleagues so they can build upon these new insights. Alternatively, it might be necessary to communicate scientific discoveries with the general public. Since different audiences are targeted in these two examples, there are also different requirements for the communication to be effective. In the case of sharing findings with colleagues, a live presentation using a simple slide show might be appropriate to show the results. For a broader audience, it is necessary to explain the discoveries along with their context, while still remaining interesting and engaging for a non-expert. In this case, it might be fitting to produce a data video that is watched by that audience on their own. These simple examples show that communication in visualization workflows is a multifaceted task. Moreover, communication is the most time-consuming part of the whole analytical workflow [TC06]. Nevertheless, research focusing on this aspect of visualization has only gained popularity in recent years [Fig14]. In particular, the concept of combining *storytelling* with visualization has gotten attention, especially within the information visualization community. The basic principle of storytelling is to convey facts by consolidating them as a *story* or *narrative*. A story is "a chain of events linked by cause

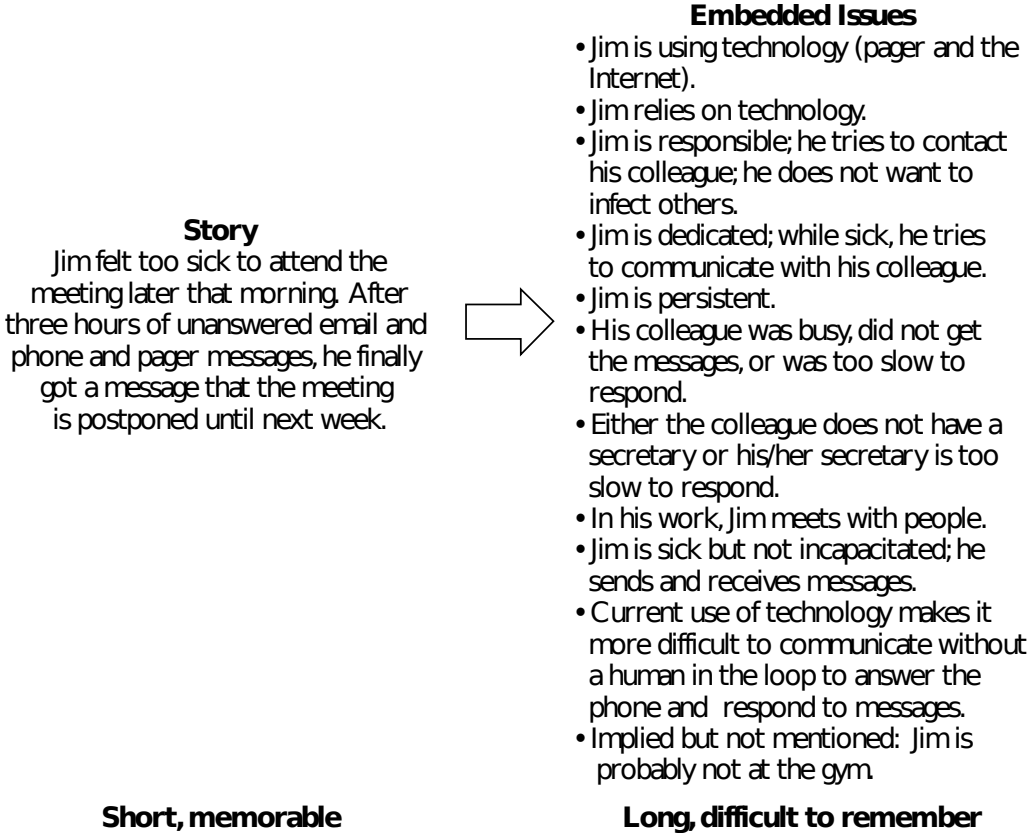


Figure 2.1: Example of how a story can contain plenty of implicit information while remaining short and memorable.

Source: Gershon and Page [GP01]

and effect and occurring in time and space" [TB03]. The main components of a story are the events or facts that are causally linked. They can be either presented explicitly or inferred [TB03]. Thus, a lot of information can be included implicitly within a story in an efficient manner. As a result, stories are one of the most efficient and memorable ways to convey information [KM13] and have been used extensively throughout human history [LHV12]. Figure 2.1 shows an example that demonstrates the advantages of communicating information as a story over simply listing facts.

Gershon and Page were one of the first to investigate the application of storytelling principles in visualization back in 2001 [GP01]. They show how knowledge about storytelling in literature and film can be used to create compelling presentations of visual data. Despite the work of Gershon and Page, storytelling remained mostly insignificant within the visualization community until the topic was revisited by Segel

Figure 2.2: Bay Model exhibit at the Exploratorium in San Francisco. Visitors can interact with the model to learn about the dynamics of San Francisco Bay's currents and tides.

Source: Ma et al. [MLF+12]

and Heer in 2010 [SH10]. The authors investigate the design space of storytelling in visualization by examining visualizations from various sources. The majority of the analyzed examples stem from online articles of newspapers. Online journalists regularly make use of interactive visualizations in their articles to make their text- and image-based stories more engaging [SH10]. Since then, many others have discussed the application of storytelling in visualization, including detailed design studies [KM13, HD11, HDR+13, Fig14, LRIC15, AHRL+15, TSS+18]. Ma et al. investigate the challenges and opportunities of storytelling in scientific visualization [MLF+12]. They present the Scientific Visualization Studio (SVS) as an example of how storytelling can be applied successfully in scientific visualization. The SVS works closely together with researchers at NASA to produce high-quality visualizations regarding NASA's ongoing research activities. Science museums are another example of practical application of storytelling in scientific visualization. Figure 2.2 shows an exhibit from the Exploratorium in San Francisco, visualizing the tides and currents of San Francisco Bay interactively. Tong et al. present a survey about storytelling in visualization [TRL+18]. They classify the literature

according to what aspect a work focuses on mainly. This classification shows that some aspects of storytelling are largely unexplored so far. These include the issues of measuring user engagement, data preparation, and — especially relevant for this thesis — storytelling for scientific visualization. The rest of this chapter discusses the design space of visual storytelling in more detail and presents relevant practical examples.

2.1 Visual data stories

In the previous section, we defined a story as a set of events and facts that are structured as chain of causally related elements. In the context of visualization, these elements include visualizations of the data that are analyzed. Stories that consist mainly of data visualizations are called visual data stories [LRIC15]. However, such a general definition is not precise enough for a detailed discussion of storytelling in visualization. Lee et al. point out that such ambiguous definitions may also include simple visualizations with little additional information or structure [LRIC15]. They propose a narrower definition that includes three basic requirements for a visualization to be considered a visual data story [LRIC15]:

1. The elements of a visual data story must be facts supported by data. That is, the information has to be derived from data visualizations.

2. These visualizations are augmented by annotations to point out the interesting parts of the data. Annotations may include text labels, arrows, highlights or narration. The purpose of these annotations is to emphasize the message that the story tries to convey. Thus, multiple visualizations without any further explanation of the data do not constitute a visual data story.

3. Finally, a visual data story must have a narrative structure that helps conveying its message. In other words, it has to be a story according to the general definition, that is, a chain of causally related elements.

Even when considering this narrow definition, visual data stories come in different shapes and formats. Segel and Heer analyzed 58 visualizations and developed a taxonomy for visual data stories based on their observations [SH10]. They identify seven different genres of visual data stories:

Magazine style
Stories of this genre are bodies of text augmented by visualizations. These kinds of data stories are common in online journalism [SH10]. The narrative structure is defined by the text, the data shown in the visualizations supports the plot. The visualizations might be static or offer interaction for the reader to explore the data.

Annotated charts
This story type is similar to the magazine style type, however the narrative is

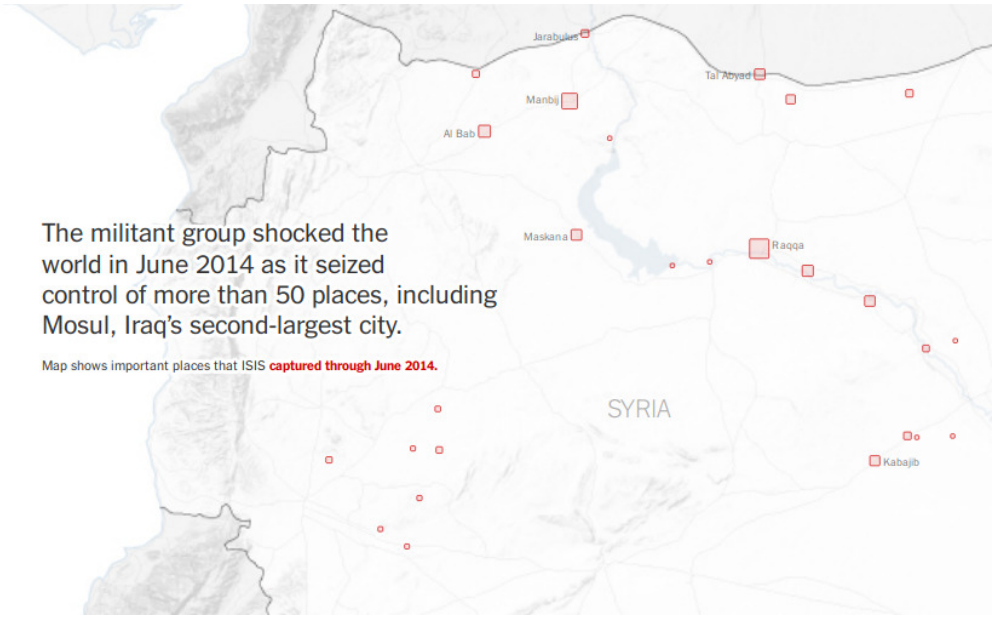


Figure 2.3: Example of an annotated chart in an online article of The New York Times. The article shows the rise and fall of the Islamic State in Syria. The background shows an annotated map of Syria. The reader can progress through the story by scrolling, which updates the map and text annotations.

Source: Almukhtar et al. [AGLW17]

mostly driven by the visualizations. Text annotations explain the main points seen in the data visualizations. Figure 2.3 shows an example from an article in the New York Times regarding the Islamic State's control in Syria throughout the years.

Partitioned posters

Partitioned posters are structured similarly to posters at science fairs [SH10]. The information is distributed among several self-contained sections, which explain some aspect of the story. The author does not prescribe a strict order in which the story has to be read. Instead, the attention of the reader is directed by highlighting and elements like arrows or lines. Apart from these guides, the reader can explore the story at their own accord.

Flow charts

Flow charts structure the story as a process. They have designated starting and end points, leading the reader through the story according to a predetermined path. There may also be decision points where the reader can decide themselves which path to explore next. Figure 2.4 shows a flow chart of a model used to determine the voting preference of a US voter based on factors like race, religion, and income.

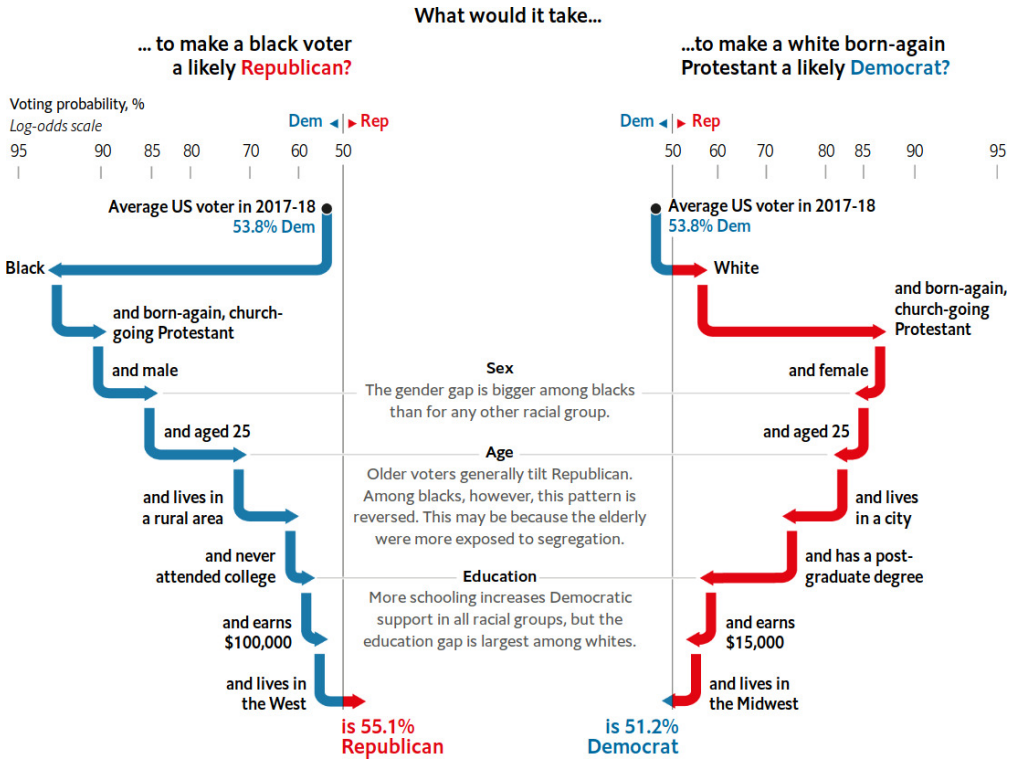


Figure 2.4: Flow chart model showing how factors like race, religion, and income impact the political party preference of US voters.

Source: The Economist [The18]

Comic strips

Comic strips are similar to partitioned posters in that the story is divided into different sections. For comics the story is structured in panels that contain images and text [McC06]. The main difference to partitioned posters is that the order is set by the author. The reader cannot choose to explore the story in a different order.

Slide shows

Slide shows also split the story in separate parts. Here, the facts are presented as slides that are shown one after another. This form of story is suitable for synchronous storytelling scenarios, in which a presenter is responsible for delivering the story to the audience. It is also possible to let the audience experience the story on their own. In this case, the reader controls when the story progresses to the next slide. Slide shows can either be completely linear or include branches like flow charts, giving the reader more control over the order of the story.

Animations

Animations allow for the most one-sided way of storytelling. The audience experiences the story in the order and pace that is determined by the author. This increases the flexibility for creators to choose from a diverse set of narrative structures for their stories [AHRL+15]. Due to the non-interactive nature of this format, it is possible to reach a broad audience, for example by producing and publishing a video to a streaming platform.

Even though this taxonomy distinguishes between seven distinct genres, there may be visual data stories that fall between two or more of the classes. It is also possible that a story incorporates multiple genres at the same time. For example, the story shown in Figure 2.3 starts off as an annotated chart. However, as the reader reaches the halfway point of the article, it continues with a magazine style format. The choice of story genre to use to convey a message, depends on the specific storytelling scenario. For example, magazine style stories are especially suitable for online newspapers, allowing for a high degree of reader interaction with the story. Live talks usually rely on slide show presentations. Section 2.4 explores the most common scenarios in more detail.

2.2 Storytelling workflow

Integrating storytelling into a data analysis workflow requires the transformation of findings into a story. Figure 2.5 outlines this process using a simple, semi-linear model. It shows the workflow as well as the information flow, that is, the transformation of raw data into a finished story.

The workflow can be divided into three consecutive stages, which process the data in five different states [GLG+16]:

Exploration

During the exploration stage the raw *data* are analyzed to discover relevant *findings*. The findings are the main content of the final story but do not have a useful shape or form yet. In order to be included into the story, *artifacts* have to be produced from the findings. Artifacts summarize the findings and can be in various formats. For example, this may include plots, charts, or images that show the main facts. However, more complex artifacts such as 3D models might also be the result of the production step.

Authoring

The authoring stage entails the actual creation of the story, which summarizes the findings that are to be communicated. To this end, the artifacts are compiled and ordered in a way that the information is presented in a meaningful and memorable way to the intended audience. This results in a *plot* that determines the structure of the final story. The *story* is the manifestation of the plot and usually belongs to one of the genres described in Section 2.1. It is used to present the information to

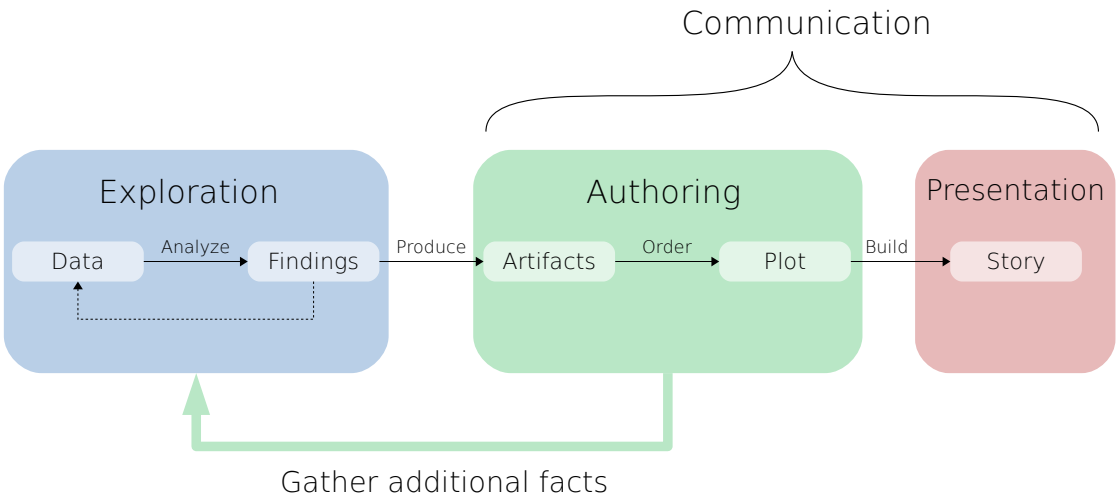


Figure 2.5: Simple model of the traditional visual data exploration and communication process based on the work of Lee et al. [LRIC15] and Gratzl et al. [GLG+16]. It shows the data flow as well as the three stages of exploration, authoring, and presentation. Communication deals with the two latter stages.

the audience. As already mentioned, the genre of the story should be chosen based on the specific storytelling scenario.

Presentation
In the presentation stage the story is shown to the intended audience. The presentation of the story can occur in various scenarios, which are discussed in more detail in Section 2.4.

Note that, while the model is mostly linear, it is possible to return to the exploration stage if more facts are required during the authoring stage. Section 2.5.6 describes another model that allows for more flexible stage transitions.

There are various roles responsible for the different steps of the transformation of data into a story [LRIC15]:

Analyst
The analyst is responsible for exploring the raw data and discovering findings. Obviously, this role has to be taken by a domain expert who is familiar with the visualization of the data. The analyst also has to produce the artifacts, which are used in the story.

Scripter
The scripter writes the plot; that is, they order the artifacts and decide on a narrative structure. A scripter has to be experienced in storytelling to create a plot

that is interesting and engages the audience. The scripter also has to work closely together with the analyst in case more information is required to build a cohesive plot.

Editor
Building the final story is done by the editor. The artifacts have to be transformed into a presentable form in accordance with the plot. Depending on the final form of the story, this role might require expert knowledge such as video editing.

Presenter
The task of the presenter is to deliver the final story to the audience. This role might not be required if the story is self-running. For example, data videos are consumed by the audience on their own; a presenter is not necessary in this scenario.

A role might be taken on by multiple people, so there might be more than one analyst, scripter, editor, and presenter working on a story. It is also possible that a single person is responsible for multiple roles. A domain expert who wants to communicate their findings with colleagues is probably going to play all the roles at once. In a more professional setting like the SVS and NASA, the roles are shared between teams of multiple people.

2.3 Interactive storytelling

Interaction in narrative visualization denotes the ability of the consumer of a story to direct its flow to some degree. It increases a visual story's comprehensibility, credibility, and involvement of the audience [MLF+12]. A consumer may pause the story and view the data from various point of views, which helps its comprehensibility. Enabling interaction especially helps the user to understand the spatial context of 3D data [WH07]. Interactive visualizations are more credible because the consumer can verify the story points within the data. Finally, interaction leads to more participation and involvement of the audience in the story and its data; hence, making it more memorable. Segel and Heer distinguish between author-driven and reader-driven visualizations [SH10]. Author-driven visualizations do not incorporate any interaction from the consumer, while the author defines the path of the story. This gives the author full control over the narrative structure, making it more efficient to convey a specific message [SH10]. In a reader-driven approach the author does not impose a strict narrative order. This lets the consumer explore a visualization freely but prevents the author from conveying a message. Due to this lack of messaging, reader-driven visualizations cannot be visual data stories by the definition from Section 2.1. Yet, it is possible to integrate user participation into a story with an author-defined structure. At first glance this may seem contradictory, as the interaction by the reader might divert the story from its predefined path. This is commonly known as the narrative paradox [MLF+12]. For visual storytelling this paradox can be resolved by either (1) alternating between an author-driven and a reader-driven mode at specific points in the story, or (2) constraining the interaction to parameters that

do not influence the narrative structure [WH07]. Based on this, Wohlfart and Hauser distinguish between four types of story consumption in narrative visualizations [WH07]:

Passive story consumption
This setup is equivalent to author-driven approaches. The reader does not have any interaction possibilities with the story, which is played back at a pace that is set by the author.

Story playback with interactive approval
The story playback is automatic but stops at certain points, giving the user an opportunity to interact with the visualization. They might change the view, representation or the content of the data for means of exploration. Once the consumer is satisfied, the control is given back to the story and it continues where it originally left off.

Semi-interactive story playback
This form of consumption is similar to story playback with interactive approval. The difference is that the observer may opt to return to story playback, while retaining some changes made during the interaction phase. For example, the user may change representation parameters of the data and then view the rest of the story using these modified parameters.

Total separation from story
The consumer can disconnect from the story completely, exploring the data on their own. Often, the switch to an unguided mode happens at the end of the story. This particular form is called the *martini glass structure* [SH10]. The stem of the glass represents the single author-prescribed path; as the glass widens, that path is left and opens up for unguided exploration of the data.

The allowed level of interaction can be categorized in regards to what parameters of the visualization can be modified. Wohlfart and Hauser distinguish between viewing, representation, and content interaction [WH07]:

Viewing interaction
Interaction is limited to viewing parameters. Thus, a user may neither change what part of the data is shown, nor how it is shown. Viewing parameters can be freely modified, e.g. the position and orientation of the virtual camera. In a 3D context, viewing parameters may also include lighting.

Representation interaction
The representation of the data can be adapted; hence, the user can determine how the data are visualized. For example, color mappings or rendering modes may be adjusted. Still, the data themselves cannot be changed.

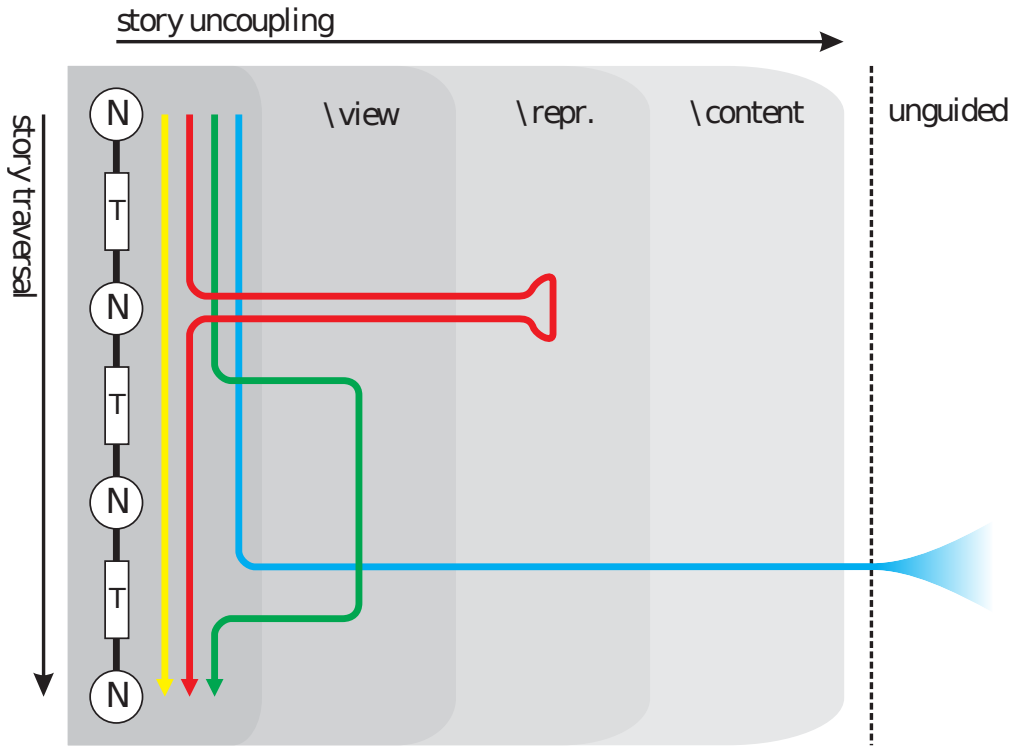


Figure 2.6: Examples for the four types of story consumption: Passive story consumption (yellow), story playback with interactive approval (red), semi-interactive story playback (green), and total separation from the story (blue). A story consists of nodes (N) and transitions (T) between them. The x-axis denotes the allowed level of interaction with the data.

Source: Wohlfart and Hauser [WH07]

Content interaction

The user interacts at the data level, i.e. they can alter what is visualized. For instance, data might be filtered or transformed to gain new insights. Even additional datasets may be loaded into the visualization. This type of interaction diverts the visualization state farthest from the narrative.

Figure 2.6 shows an example for each of the story consumption types with varying permitted levels of interaction. The example for story playback with interactive approval (red) makes use of viewing and representation interaction. The semi-interactive story playback (green) is limited to viewing interaction in this case.

2.4 Scenarios

Storytelling can be applied in different scenarios and settings, which have varying requirements. These requirements affect every stage of the storytelling process [LRIC15]. For example, a non-expert audience is unlikely to get captivated by a story presenting overly detailed domain knowledge without further explanation. This has to be considered when selecting findings to present, when forming the plot, and during the actual presentation. Aside from the intended audience and their expected background knowledge regarding the domain, a scenario also involves how the story is delivered. Traditionally, presentations involve a speaker talking to an audience. However, visual data stories can also be self-running, i.e. there is no presenter responsible for delivering the story. For instance, videos published on a streaming platform or online newspaper articles fall into this category. Lastly, the scenario can also be characterized by (1) if the audience can interact with the story during its presentation, (2) how they can interact, and (3) to what degree they can interact (see Section 2.3). Based on these observations, three common storytelling scenarios can be identified [KM13]:

Self-running presentation

In this scenario the story is consumed by the audience on their own; there is no presenter delivering the story. This scenario is especially suited to reach a wide audience, since each consumer can watch or read the story independently. A disadvantage is that independently watching or reading consumers may be inclined to stop paying attention, if there is a lack of interest and engagement. Therefore, it is a major concern of self-running presentations to catch an initial interest of the user [KM13]. Stories in this scenario can vary in what type and level of interaction is provided to the audience. Interactive self-running presentations are especially common in online journalism [SH10]. NASA and SVS produce animations and videos, which are examples of passive forms of self-running presentations.

Live talk

Live talks are scenarios that are common in business presentations [KM13] and conference talks. The story is delivered by a presenter to an audience that may consist of up to a few hundred people. During the presentation itself there is little or no participation from the audience. Hence, the audience does not have an immediate impact on the story. Nevertheless, there is usually a discussion session after the presentation, allowing the audience to ask questions that are answered by the presenter. To this end, it is desirable that the story supports some level of interaction, so facts can be revisited and viewed with different parameters.

Dynamic discussion

This scenario is similar to live talks, as a presenter delivers the story to an audience. The main difference is that the audience is small and actively participates during the presentation. The audience usually consists of domain experts in this scenario. Questions are raised and discussed during the presentation, enabling the story to

divert from the originally planned path. As a consequence, it is vital that the story allows for interaction up to the content level.

2.5 Examples

This section discusses a few practical examples of storytelling in visualization. These include applications in both scientific and information visualization.

2.5.1 Storytelling for volume visualization

Wohlfart and Hauser present a design and implementation of storytelling in volume visualization [WH07]. Their stories consist of nodes, connected by transitions. A node is a visualization state in the analysis process, augmented by text annotations. Two types of annotations can be added to a state: (1) general text labels centered at the bottom of the view, and (2) text describing specific features of the dataset. The former act as captions describing the whole scene, which is common in medical applications [WH07]. Figure 2.7 shows an example of how both annotation types are applied in practice. A transition between two nodes ensures continuity and enables consumers to build a mental map of the data. Each transition consists of multiple action groups, which are played back consecutively. A duration can be assigned to each action group, giving the author exact control over the execution of a transition. The actual changes to the scene are represented by action atoms, which are contained in the action groups. The atoms are played back simultaneously within a group and control one aspect of the visualization each. For example, there might an atom controlling the view and another one changing the transfer function. The final story is an animation, guiding the viewer through the relevant stages in the analysis process. Thus, the story nodes can be seen as animation keyframes with the transitions determining the interpolation between them. In order to enhance the viewer's understanding of the spatial relations within the story, playback allows for interactive approval up to a viewing interaction level (see Section 2.3).

Authoring a story is achieved by explicitly recording the interaction of the user with the visualization. Once the user has enabled the recording mode, every input such as changing the view or representation parameters is saved in a story action group. This way a first draft of the story can be generated. Then the story can be edited by selecting a story node or transition via the graphical user interface (GUI). At this point, more actions can be recorded and inserted at the selected position. Hence, stories do not have to be recorded as a single, continuous sequence of interactions. Feature-specific annotations are manually positioned in 2D screen space by the user, but also refer to a position in 3D data space. The 3D position is selected by clicking into the scene. A corresponding picking ray is intersected with the data, returning the anchor point of the annotation. Dragging the mouse controls the size of the circle of the arrow.

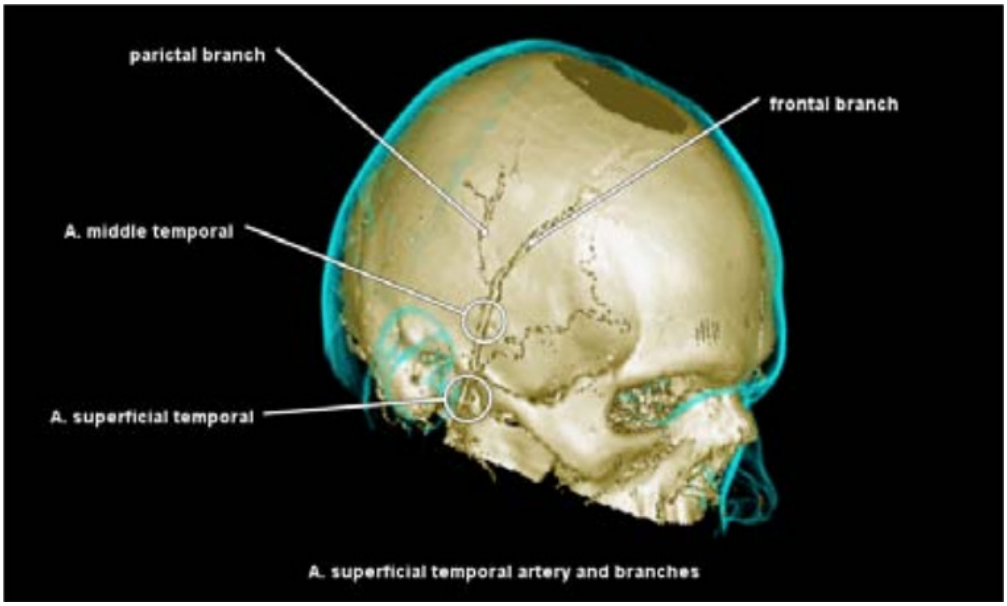


Figure 2.7: Text annotations can be added to a visualization state. The labels can either act as a caption to the whole scene, or describe specific features by means of an arrow pointing to the corresponding position in the data.

Source: Wohlfart and Hauser [WH07]

2.5.2 AniViz

Akiba et al. also discuss the addition of storytelling mechanisms in volume visualization [AWM10]. Their system lets users build non-interactive animations for time-varying, multivariate data based on a set of predefined templates. Instances of these animation templates can be combined to produce complex animations, giving the creator great flexibility. The templates correspond to basic motions usually found in visualization animations and consist of a set of parameters. Specifying the parameters of a template yields an instance thereof. The following templates are available [AWM10]:

Overview
: Animates a set of parameters (spatial, temporal, variable, or transfer function) according to a default path. For example, a spatial overview instance results in a 360° rotation of the data along the y-axis.

Spatial exploration
: Animates between two given camera orientations according to a given interpolation function.

Temporal exploration
: Animates between two time steps using a given playback function.

Variable exploration
Changes a variable from one value to another using a blend function.

Transfer-function exploration
Interpolates between two transfer functions given an interpolation function.

Highlighting
Changes parameters of a specific object according to a periodic function. The authors showcase this template by highlighting a tumor in a Magnetic Resonance Imaging (MRI) scan dataset of a human head. In this case, the template is used to vary the opacity of the tumor periodically.

Slicing
Clips the data to reveal internal structures.

Image
Adds images to a transition.

Caption / Annotation
Adds text annotations to a transition.

Alternatively, the user can also directly specify keyframes for the animation. In this case, AniViz automatically computes the appropriate template instances from the given keyframe and its context. Multiple instances can be combined by either parameter-space or image-space blending. If the instances do not overlap on the temporal axis, the first and last frame are interpolated respectively to compute an intermediate frame. If the instances overlap temporally, the corresponding frames are blended together. For parameter-space blending, the affected parameters are blended. Image-space blending combines the corresponding image frames directly on a pixel basis. This type of blending templates can be used to generate fade-in and fade-out effects.

Figure 2.8 shows the GUI of AniViz. It consists of three parts: the explorer, the viewer, and the mixer. The explorer contains the interface elements that are used for the analysis stage. In this example, there is the render window along with two windows for modifying the parameters of the visualization. The mixer provides controls to build the story by instantiating the displayed templates. It also shows the current story as multiple tracks of template instances. Each instance is displayed as a rectangular, colored widget. Its position along the x-axis determines its temporal position in the animation. The distribution among the tracks influences how the instances are combined. Image-space blending is applied whenever two instances are aligned in two different tracks. If an instance is placed in between two tracks, parameter-space blending is applied (e.g. the highlight instance in Figure 2.8). Naturally, instances can be moved, edited, and deleted after they have been added to the story. The viewer shows the current story as a sequence of thumbnails, which are called keys in this case. It also provides controls for playing, pausing, and stopping the animation. Moreover, the viewer can be used to control

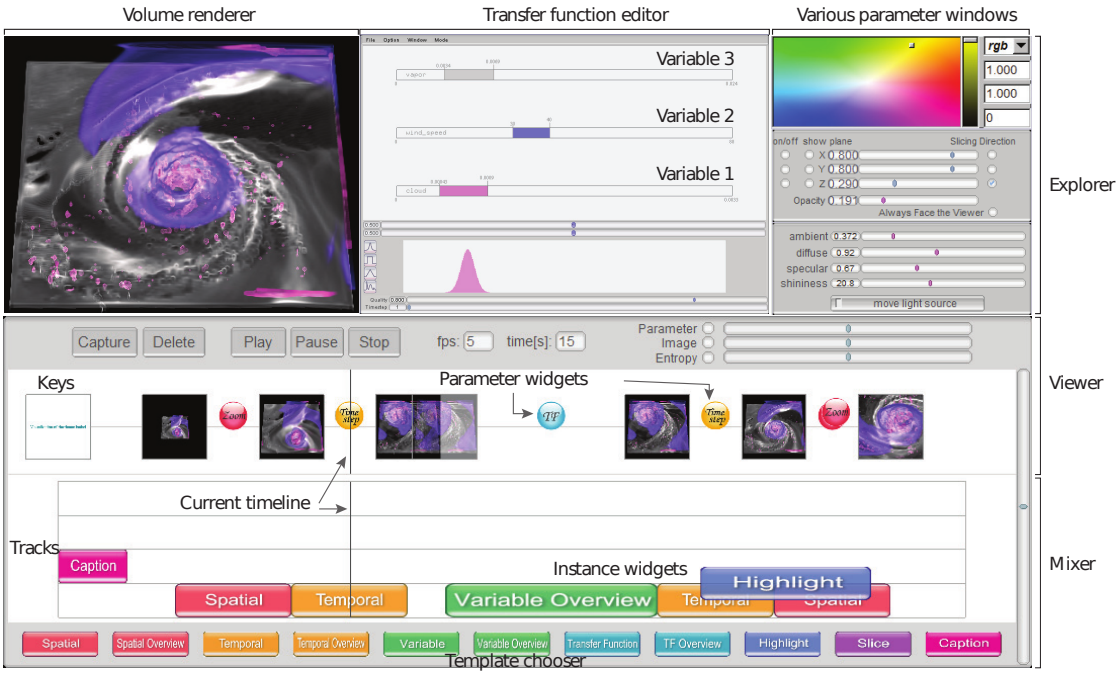


Figure 2.8: GUI of AniViz. It consists of the explorer, viewer, and mixer.

Source: Akiba et al. [AWM10]

the duration of transitions between keys. The duration can be either set manually, or automatically based on three distance metrics: parameter, image, and entropy.

2.5.3 VisJockey

VisJockey is a storytelling approach by Kwon et al. that aims to facilitate the creation of magazine style stories [KSJ+14]. In particular, the focus lies on journalists who incorporate such visual data stories in their online articles. Often, custom and novel visualizations are put into those articles without giving the reader a proper introduction [KSJ+14]. This makes it harder for the reader to validate the main points of the textual article in the visualization. VisJockey tries to alleviate this problem by enabling authors to define so-called plays. Plays trigger certain actions that are supported by the visualization (e.g. zooming, panning) as well as additional effects such as highlighting, annotations, and animated transitions. They can be linked to text segments in the article and triggered by the reader. This integrates the visualization tightly into the article, and adds structure to the story. Figure 2.9 illustrates the concept with a simple example. The linked text segment is highlighted when the mouse hovers over it, and a small inline icon additionally indicates the existence of a play. As the reader clicks on the segment, the visualization triggers an animated transition. Furthermore, annotations become visible describing the data in more detail. A shortcoming of VisJockey is that there is no

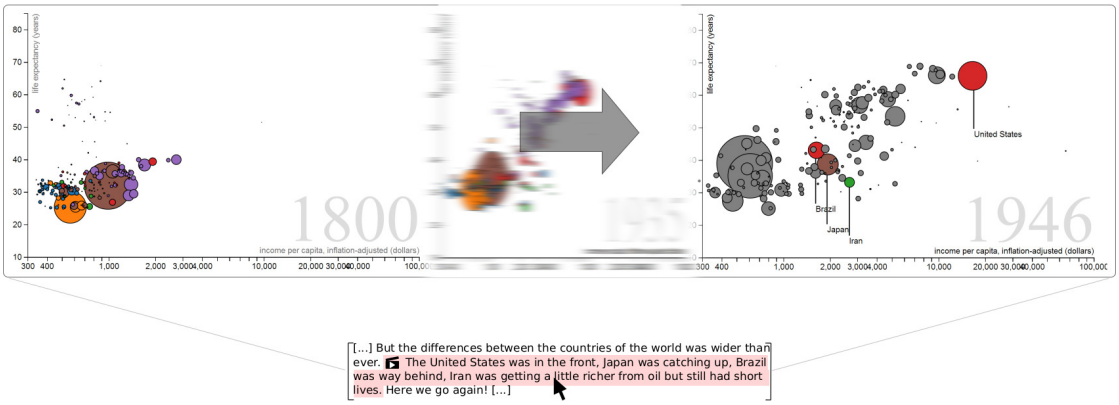


Figure 2.9: Example of how VisJockey integrates predefined visualization actions (plays) into an article. The reader triggers the play by clicking on the corresponding text segment. This leads to an animated transition and adds annotations to the visualization.

Source: Kwon et al. [KSJ+14]

dedicated authoring tool. Hence, producing plays requires programming knowledge. The authors argue that — while it is a disadvantage — online journalists are usually familiar with D3.js [BOH11], and thus should have the required programming proficiency.

2.5.4 ChartAccent

Ren et al. present ChartAccent, a tool for creating annotated charts [RBL+17]. It is possible to add manual as well as data-aware annotations to existing visualizations. Data-aware annotations consider the context of the data to be annotated so that a suitable annotation form can be selected automatically. Four forms of annotations are supported:

Text
: This type of annotation can be used to add context to the visualization or provide general comments. Data-aware text annotations display the value of an item or statistics about a set of items. For example, a text annotation can display the average of a set of targeted items. Visual properties such as font, size, and color can be chosen by the user.

Shapes
: These annotations can be further distinguished by their geometric shape. For example, it is possible to add rectangles to lead the viewer's attention to a specific region of the visualization. Other shapes include arrows, which can be used to point to a specific data element. This is particularly useful in combination with a text annotation. Data-aware shapes include trend lines for scatter plots. The appearance of these shapes can be adjusted (e.g. stroke and fill color).

Highlights
Highlights are used to modify a data element's appearance to emphasize it or divert attention from it. For example, a highlight might change an element's color or reduce its opacity.

Images
Image files can be loaded and added to the visualization. For example, a data point representing a country can be annotated by an image of its flag.

Figure 2.10 shows the GUI of ChartAccent. A specific form of annotation can be added manually by clicking on the corresponding button. The user can select data items using a bubble cursor [GB05] and create data-aware annotations by clicking on the intended target. The editor allows the user to define target selections based on simple formulas. These formulas can include simple statistical functions such as the minimum, mean, or maximum of the data values. Another way to select a target is to first select a position, line, or region in coordinate space. Then, elements might be selected relative to that manual selection. For example, the user might select a horizontal line manually in a bar chart, which corresponds to a value on the y-axis. Subsequently, all elements that exceed that value can be selected automatically. Additionally, categorical data can be selected by clicking on the category names in the legend.

The form of data-aware annotations is based on the selected target. For each type of target, a default form is defined. For example, selecting a single element or a set of elements adds a black outline, and a text annotation. Targeting the axes adds a perpendicular line and a text label displaying the corresponding value. If a set of elements is selected, a trend line can be easily added. In scatter plots this can be augmented with a bubble set annotation.

Once an annotation is added to the visualization, it can be moved via drag and drop, which also adjusts the target. Furthermore, its visual properties can be modified using the editor. This includes the text format, font, outline, and color.

2.5.5 Geological storytelling

Geological analyses of seismic data are conducted to find appropriate locations to produce natural resources like oil and gas. However, drilling test wells and gathering high-quality 3D data of a location is expensive and time consuming [LHV12]. Usually, geoscientists have to comment on the feasibility of a location based on sparse 2D seismic data [LHV12]. Seismic models are created, which aim to explain how past geological events and processes have caused the current geologic formations that are observed in the imagery. These models make it possible to estimate the probability of the existence of natural resources in the area. Based on this, a location is either dismissed, or a more involving and expensive analysis is conducted. Unfortunately, models based on these 2D seismic images are highly variable [BGSJ07]. That is, such geological analyses are susceptible to human bias, making it necessary to compare multiple models to arrive at the most plausible one.

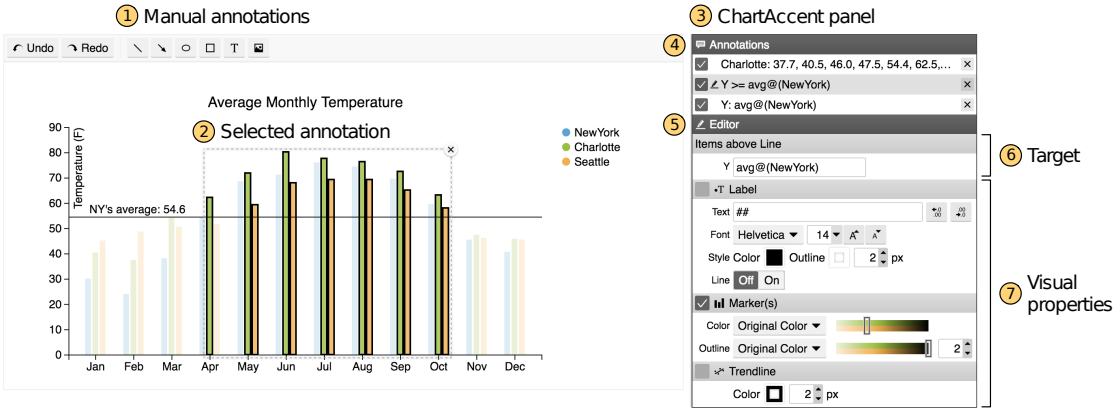


Figure 2.10: The interface of ChartAccent consists of (1) controls for adding annotations manually, (2) a dotted rectangle that highlights the currently selected annotation, (3) a panel containing controls for viewing and editing annotations, (4) a list view of all annotations, (5) controls for editing the currently selected annotation, (6) controls for modifying the annotation target, and (7) controls for editing the appearance of the annotation.

Source: Ren et al. [RBL+17]

Moreover, the first assessment on whether a location is feasible or not has to be made within a tight time constraint [LHV12]. Traditionally, geoscientists sketch models with pen and paper as it is quick and flexible [LHV12]. However, presenting models based on this medium is difficult in settings where domain experts and decision makers are situated in different locations. Digital solutions are either too complex or limited for this specific use case. Lidal et al. present geological storytelling, a tool that aims to bridge the gap between pen and paper, and digital approaches by providing a level of expressiveness similar to an analog solution [LHV12].

Geological storytelling is designed around the observation that geoscientists think in terms of geological events and processes shaping the environment. These events include the deposition of sediments, faults, and folds. A geological story captures these events and how they transform the geological layers of an area. Thus, a story in this context is a sequence of nodes representing the results of geological events. A node contains a sketch of the geological situation at the corresponding point in geological time [LHV12]. When building a model based on a 2D seismic image, the root node usually corresponds to the present situation, while subsequent nodes correspond to points in time in reverse chronological order. The sketches for these nodes are created by the geoscientist using drawing primitives supported by the tool. Figure 2.11 shows such a sketch. The 2D seismic data is shown as a background. Colored curves represent horizons, i.e. boundaries between geological layers. Horizons can be either a single, connected curve or a set of disconnected curves. In the latter case, one or multiple faults separate the horizon, indicated by vertical black curves. A geological layer is represented by one or multiple

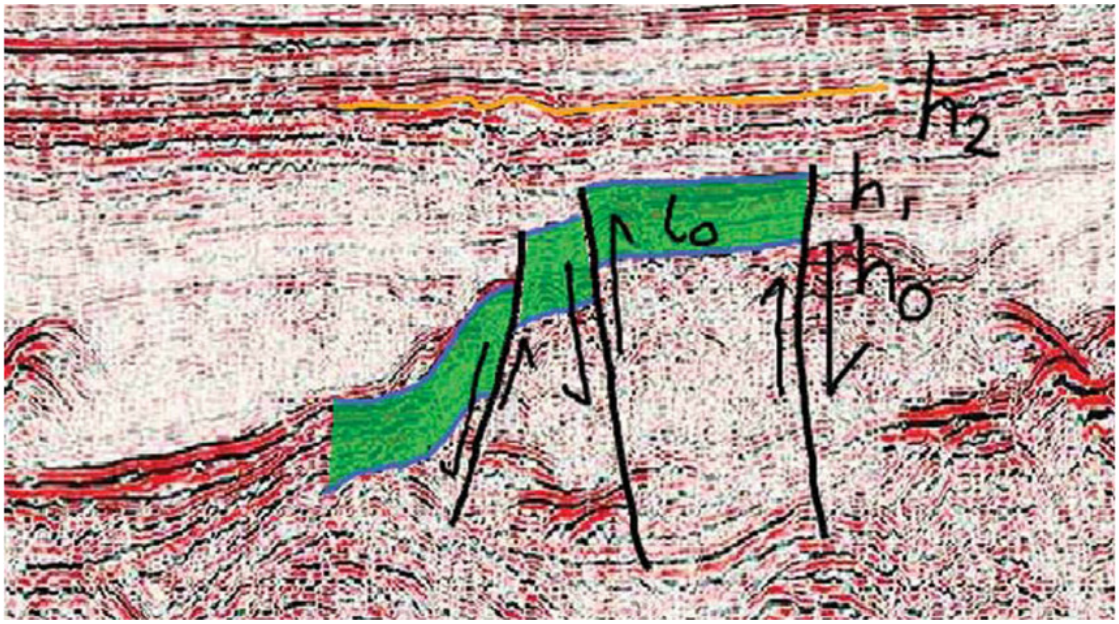


Figure 2.11: A sketch for a seismic section created in the geological storytelling. Three horizons h_0, h_1, h_2 are indicated by colored curves. The layer l_0 between h_0 and h_1 is represented by green polygons. The layer is divided by three faults, which are sketched as vertical black curves. The slip along the fault lines is indicated by black arrows.

Source: Lidal et al. [LHV12]

colored, filled polygons. Additionally, annotations can be used to indicate directions of movement, or to label elements.

Once a sketch for a story node is complete, the user may create another node by means of a flip-over metaphor. A blank page is shown for the new node, but the lines from the previous node are shown as so-called ghost lines with reduced opacity. There is usually a strong correlation between two adjacent nodes, as they represent two subsequent points in geological time. For example, a horizon line may remain unchanged between two nodes, while another one gets divided by a new fault. Thus, the flip-over metaphor facilitates the sketching of a subsequent node, and is similar to sketching in a sketch book. Moreover, it is possible to copy lines from the previous node, so it is not necessary to trace them manually if they remain unchanged. In order to create multiple models that divert at some point, geological storytelling supports branches in stories. The user may return to a previous node in the story (or even the root node), and create an alternative successor node. Thus, a branch is created, which represents two stories that are identical up to this point and then divert. This results in a story tree, which is displayed in the GUI for navigation.

In order to communicate a completed story, an animation can be generated from it

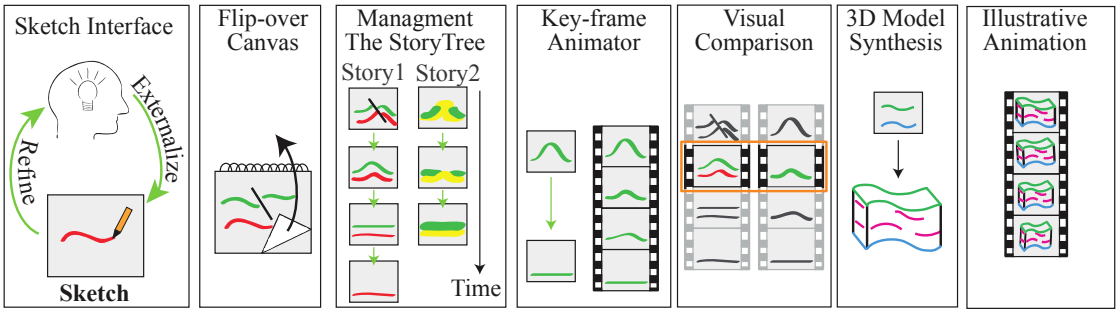


Figure 2.12: Workflow of geological storytelling. The user sketches story nodes on a flip-over canvas. Multiple variants of a story are managed as story tree. 2D animations can be automatically generated from the story nodes. Multiple models can be visually compared and evaluated. The most promising model can be transformed into a 3D animation in order to be presented to non-experts and decision makers.

Source: Lidal et al. [LHV12]

automatically. To this end, the story nodes act as keyframes, intermediate frames are interpolated from the corresponding nodes. The intermediate frames only animate the horizon lines, layer polygons are not interpolated and only shown during the keyframes themselves. The interpolation of horizon lines automatically handles special cases, such as the splitting of a single horizon into multiple pieces due to faults. The appearance and disappearance of horizon lines due to sediment deposition and erosion respectively is also animated in a geologically sound way. These events only happen at the top layer [LHV12], so the interpolation of the involved horizon uses the existing top horizon as the interpolation target or source. Multiple branches of the story tree can be animated next to each other, allowing for their visual comparison. To this end, the slider controlling the animation is used to play and pause both stories at the same time. Comparing multiple story variants visually side by side simplifies the evaluation of their respective plausibility. Once the most promising model has been identified, it can be used to automatically create a 3D animation. Conformal texture mapping is used to apply textures to the individual layers, which are animated in a realistic way. This form of animation is intended for presenting models to non-experts. Figure 2.12 summarizes the complete workflow of geological storytelling.

2.5.6 CLUE

The scientific method is an iterative process by nature [DC02]. Discoveries or, more specifically, theories are built upon and used as a basis for further research [DC02]. This might result in modifications to existing knowledge, or completely new insights [DC02]. Another essential aspect of the scientific process is the possibility to verify the correctness of results presented by other members of the scientific community [Ple18]. To this end, the processes and intermediate steps that lead to a finding have to be documented to an

extent that other scientists can reproduce that finding [Ple18]. Reproducing a finding entails repeating the same analyses on the same data to arrive at the same results, which is referred to as methods reproducibility by Goodman et al. [GFI16].

The traditional storytelling workflow model presented in Section 2.2 and Figure 2.5 does not explicitly reflect these two characteristics of the scientific process. It is semi-linear and ends with the presentation stage. For true methods reproducibility, however, a transition from the presentation stage to the exploration stage is required. This way a consumer of the story can easily retrace the exact steps that lead to discoveries conveyed by that story. Similarly, building upon findings is facilitated by a backlink to the exploration stage. The consumer may view the analysis process of a story, modify it and extend it. Therefore, revising the traditional storytelling workflow is necessary to accommodate for these requirements. Gratzl et al. propose Capture, Label, Understand, Explain (CLUE), a workflow model that includes additional transitions from the presentation stage to the other two stages [GLG+16]. Figure 2.13 shows a visual representation of the CLUE model. Apart from the stage transitions, CLUE also adapts the information flow of the workflow. In the traditional workflow, findings are transformed into artifacts, condensed into a plot, and finally included in the final story. This sequential data model does not integrate with the iterative workflow, since it is generally not possible to return from artifacts (e.g. screenshots) to the original findings. In other words, it is not possible to conduct additional analyses based on artifacts alone. CLUE overcomes this limitation by basing its data model on a *provenance* system. Provenance is the lineage of a data item, i.e. metadata that describe how that specific datum was derived [SPG05]. In the case of CLUE, the provenance of the entire analysis process is captured automatically. Hence, the recorded provenance describes how a specific visualization state was achieved. These data are saved as a provenance tree, where a node represents an intermediate visualization state and an edge represents the action that lead from one state to another one. Branches represent alternatives in the analysis process that were explored by the user. Provenance makes it possible to navigate to any visualization state that was explored during the analysis process. Finished stories save and reference this provenance information, so it is possible to go back to the analysis and authoring stages. Figure 2.14 shows two exemplary workflows that are supported within the CLUE model. Figure 2.14a represents the traditional semi-linear storytelling workflow. It starts with an iterative analysis session in the exploration stage, and continues with the authoring and presentation of a story conveying the main findings. Figure 2.14b is a workflow that is based on the idea of building upon previous discoveries. It starts in the presentation stage with consumers reading or listening to a finished data story. They use this story as a starting point for their own analysis and switch back to the exploration stage. At this point, they can retrace the analysis process captured in the original story and explore the data further. If new discoveries are made, the workflow continues with the authoring and presentation of a story including these discoveries. Thus, the starting point of the workflow is reached and starts over with a consumer of that second story.

Visual data stories in CLUE, also called *vistories*, are slide shows that summarize the

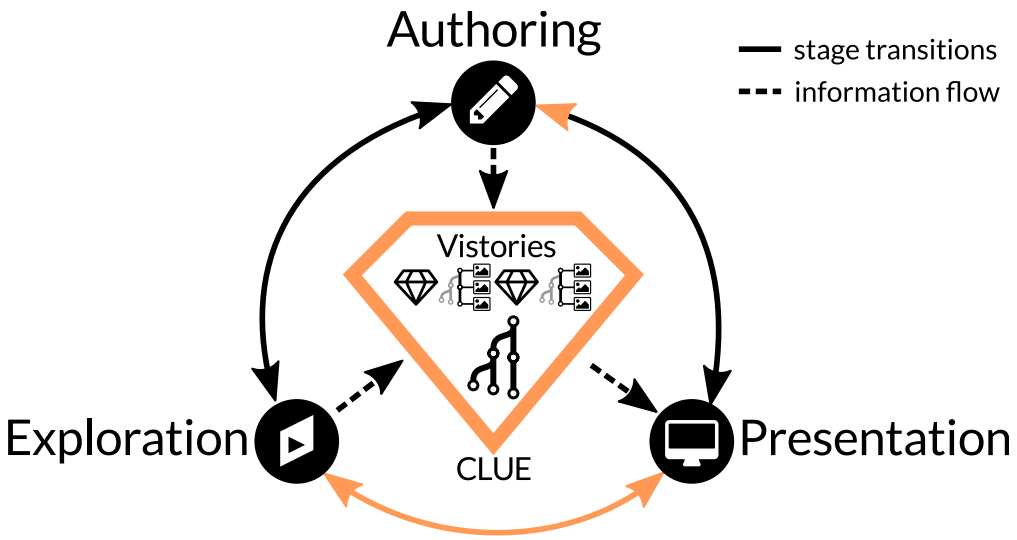


Figure 2.13: The CLUE model for storytelling workflows. It extends the traditional approach (black arrows) by allowing arbitrary transitions from the presentation stage (orange arrows). Its data model is centered around interactive, visual data stories called *vistories*, which are based on provenance data.

Source: Gratzl et al. [GLG+16]

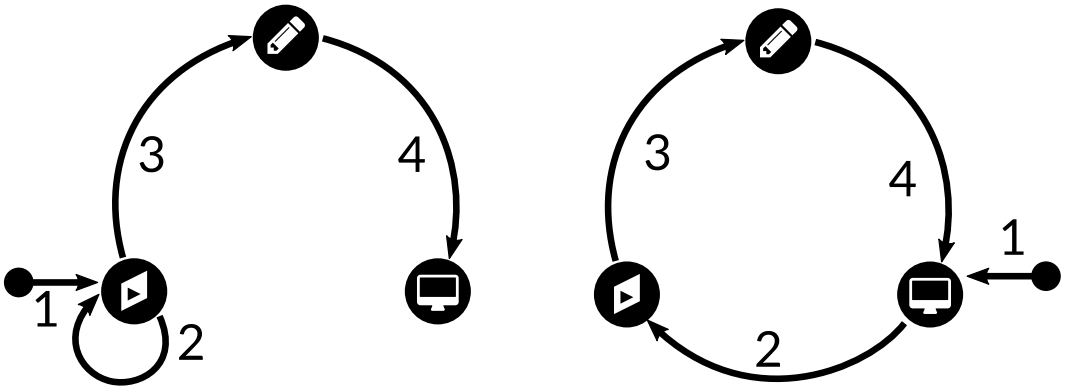


Figure 2.14: Two examples of workflows in the CLUE model. The traditional approach starts with an iterative exploration step (a), while it is also possible to start at the presentation stage and build upon an existing vistory (b).

Source: Gratzl et al. [GLG+16]

analysis process. Each slide either consists of just text for titles and captions, or references a node in the provenance tree. The latter slide type represents a visualization state and can be enhanced with explanatory annotations. Thus, vistories are essentially linear paths through the provenance tree. Vistories are created and viewed within the same tool that is also used for the analysis. Every slide can be assigned a viewing duration before the next slide is shown, so the progression through the story is automatic. Alternatively, the presenter or reader controls the pace at which the story unfolds. Transitions between two state slides are animated, so the viewer can relate the two visualization states more easily.

CLUE is implemented as a library for Caleydo Web, a visual analysis platform for biomedical data [GGL+15]. The library functions are called by the visual exploration tool and handle the capturing and processing of provenance. The library provides a GUI for viewing and managing the provenance data as well as the stories. Figure 2.15 shows the GUI of the prototype implementation of CLUE. The provenance window displays the provenance tree as a node-link diagram. It follows a vertical layouting algorithm that keeps the node of the current state and its ancestors in a straight, vertical line. Animated transitions are used to visualize layout changes when another node is selected. Nodes are displayed in four varying detail levels, which are chosen according to a Degree of Interest (DoI) function. The DoI function depends on several factors, such as the distance to the current node, if the node is along the main path, and if the node is bookmarked. At the lowest detail level, a node is displayed as a bullet point. As the detail level increases, icons for the corresponding action, labels describing the state, and thumbnails of the visualization state are displayed as well. The nodes of the provenance graph cannot be edited directly, since this would falsify the metadata. Instead, the user may navigate to any node and perform an action that results in a state that is different from its original successor. This spawns a new branch and represents an alternative path in the analysis process. Branches are positioned on the left of the current path, leaving space for the detailed descriptors on the right. Actions of a branch can also be replayed on a different branch by dragging and dropping that branch on a node. This allows for an easy creation of a copy of a branch that differs in a single action.

The story editor displays the current vistory and provides controls for adding, editing, and removing slides. Similarly to the provenance tree, slides are displayed in a vertical fashion starting from the top. The height of a slide indicates its duration during automatic playback, which can be adjusted via dragging. State slides can be created by (1) extracting the current state and all its ancestors, (2) extracting all bookmarked states, or (3) manually using the new slide button. The latter option creates a new slide based on the current state. Existing slides can be reordered using drag and drop. Three types of annotations can be added to the slides: text, arrows, and rectangles. These are positioned in the coordinate space of the visualization (e.g. relative to a data point), and thus remain valid when the aspect ratio or screen resolution changes.

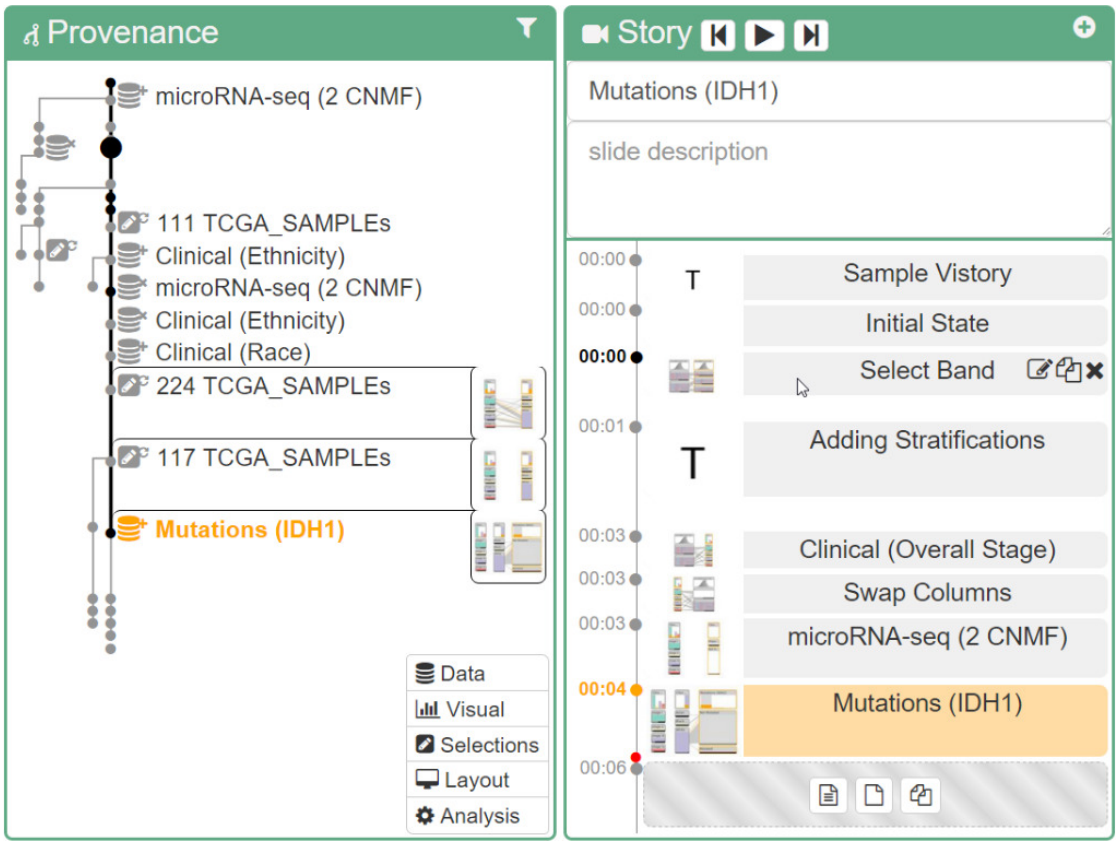


Figure 2.15: Windows in the CLUE GUI showing provenance and the current story respectively. Provenance is displayed as a tree with multiple LoDs. The slides in the story editor are arranged vertically, where the height of a slide designates its duration.

Source: Gratzl et al. [GLG+16]

CHAPTER 3

Provenance

Provenance is the lineage of a data element [SPG05]. It comprises all the information that describes how the datum is derived, and in turn what other data are derived from it. Provenance is essentially a graph that describes the dependencies between the elements of a process, i.e. sources, computation steps, and the output [CAB+14]. Thus, the interdependence between data and processing steps is preserved, which makes it possible to pose and answer questions such as [CAB+14]: Where does a datum come from? How is it obtained? Who is using it? A common use case for provenance is the exact replication of previous experiments and results [RESC16]. It is also possible to answer more elaborate questions that are not directly evident from the graph. For example, comparing the provenance of a correct and an erroneous run of an experiment, makes it possible to find the origin of the errors in the latter [CAB+14]. For this work, provenance is captured for the whole analysis process conducted in PRo3D. A node in the graph corresponds to a visualization state, while edges represent user interactions that lead from one state to another one. The metadata of a node contains versioning information, so the user may return to a previous visualization state at any time. Similar to CLUE (see Section 2.5.6), the final stories reference the provenance graph to allow for transitions from the presentation stage to the analysis and authoring stages. This chapter discusses the topic of provenance in more detail. First, it gives a broad overview of provenance beyond its specific application in this thesis. This includes a characterization of provenance based on its type and how it is employed. Then, practical examples are presented based on how relevant information is filtered, abstracted, and extracted.

3.1 Overview

The general definition of provenance given above, i.e. the lineage of data, includes a wide range of systems realized in practice. These provenance systems differ in the way how provenance is captured and represented, as well as how this information is used. They aim

to solve different problems in different domains. For example, some provenance systems focus on workflows independent of specific data. Instead, workflows are represented as recipes that may be subsequently executed with arbitrary data. Such models are known as *prospective* provenance [LLCF10]. The more common case of *retrospective* provenance entails specific data and their computational history. It is also possible to capture provenance information that encompasses both the prospective and the retrospective components [LLCF10]. This wide variety of approaches has to be taken into consideration when discussing provenance. In particular, comparing models and choosing an appropriate approach for a given problem can prove to be difficult without a systematic framework. Therefore, various taxonomies, characterizations, and definitions for provenance have been proposed over time [SPG05, CCM09, RESC16, FKSS08, NCE+11, CAB+14, GZ09]. For this discussion, we follow the taxonomy introduced by Ragan et al. [RESC16]. They classify provenance in the context of visualization according to its type and its purpose. The type of a provenance system describes what kind of information it captures, which can fall into five different categories [RESC16]:

Data / workflow

Provenance of data is concerned with the computational history of datasets. It is the most common and straightforward type of provenance, and captures all operations that transform data in some way. While some taxonomies explicitly distinguish between data-oriented (retrospective) and workflow-oriented (prospective) provenance (e.g. Cruz et al. [CCM09] and Simmhan et al. [SPG05]), provenance of data includes both perspectives in this case. Data provenance can be captured automatically at different points in the software stack (e.g. application, middleware, and operating system (OS) levels) [CAB+14]. The level of abstraction of the provenance data depends on the point at which they were captured. For example, events and messages observed at the application level inherently carry more high-level semantics than information gained by intercepting OS function calls and events. However, the former approach requires the analysis or domain application to be modified to capture provenance, while the OS level approach does not require any modifications to existing software [CCM09]. Moreover, provenance captured at a lower level tends to be of higher granularity, which may be required for some applications.

Interaction

This provenance type focuses on the history of user interactions with the application. Gotz and Zhou distinguish between three different categories of actions in visual analytic systems: exploration, insight, and meta actions [GZ09]. The exploration and insight categories are further divided into sub-categories as seen in Figure 3.1. Actions in the exploration category modify the data that are visualized or parameters of their visualization. Most user actions in visual analytic systems fall into this category [GZ09]. Insight actions are actions performed by the user when they discover new findings during the analysis (e.g. taking a note, annotating a data element). These actions can be specific to a visual object (visual insight action) or

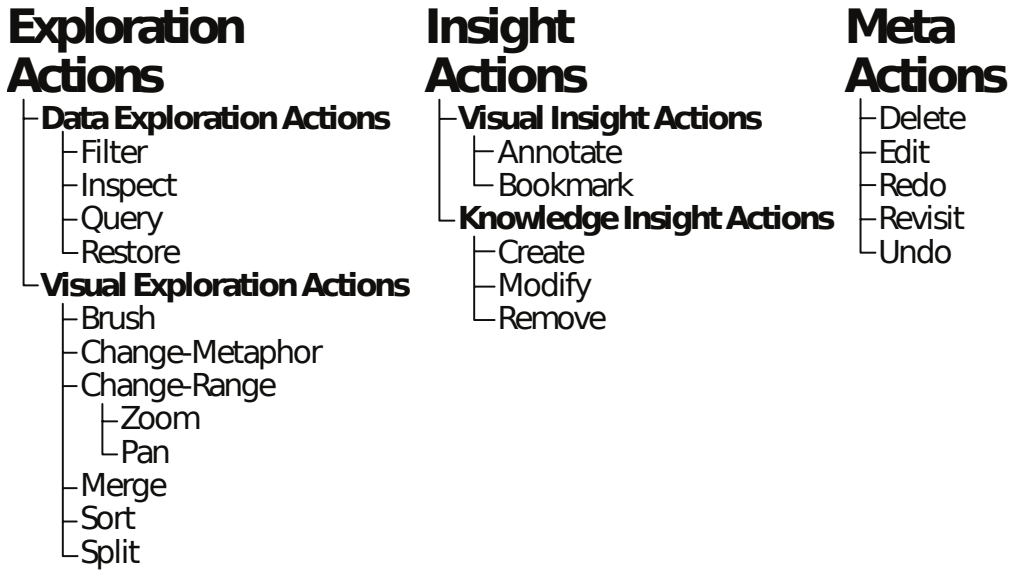


Figure 3.1: Gotz and Zhou distinguish three categories of actions in visual analytics: (1) exploration actions modify the dataset or its visualization, (2) insight actions are performed when the user discovers new findings, and (3) meta actions are actions that deal with provenance itself.

Source: Gotz and Zhou [GZ09]

general insights (knowledge insight action). Meta actions include all actions that involve the provenance itself, such as undoing an action. Interaction provenance can include all of these action categories, but is most often concerned with exploration actions. Provenance tracking meta actions is a special case also known as provenance of provenance [GLG+16].

Visualization

Provenance of visualization tracks the history of visualization states. The visualization state is obviously closely related to data transformations and user interactions, but it is possible to record visualization provenance without full data or interaction provenance [RESC16]. In a sense, visualization provenance is a subset of the latter two. Storing visualizations can be done by storing the properties of the application state that are needed to recreate those visualizations. This requires a formal model that describes those properties and derivations thereof, which represent transitions between two states [JKMG02]. Alternatively, screenshots can be recorded if the replication of prior states is not required [RESC16]. This and numerous other works are based on the former approach [GS06, HMSA08, SvW08, GLG+16].

Insight

Provenance of insight is different from the previous types as it deals with the cognitive process of the analyst. The aim is to track the history of findings or insight

that are revealed during the analysis process. North characterizes insight as complex and unexpected [Nor06]. As such, automatically quantifying and capturing insight is difficult or maybe even impossible. Therefore, insight provenance is often tracked manually by the user, e.g. by managing notes or think-aloud protocols [RESC16]. In other words, insight provenance is usually captured in combination with interaction provenance, i.e. tracking insight actions.

Rationale

Rationale is the reasoning or motivation of an analytical approach. Like insights, rationale is inherently a dimension of the cognitive process, and thus hard or impossible to quantify automatically. Therefore, provenance of rationale, i.e. the history of hypotheses and goals, may only be collected explicitly by the analyst. For example, Shrinivasan and Wijk present a visualization framework that enables the user to record their analytical process in the form of notes in a separate view [SvW08]. These notes can be linked to specific visualization states in the provenance graph, documenting the intent of subsequently taken actions. However, in contrast to insights, rationale may change more frequently, resulting in overhead for the user documenting their thought process [RESC16]. Ragan et al. also suggest that systems logs may be used to infer rationale in case the analysis process follows an expected, regular path [RESC16].

Note that the boundaries between some categories are not always clear-cut. For example, it may be difficult to separate provenance of visualization and provenance of data in some cases. As already discussed, this is due to the fact that the visualization state inherently depends on the dataset. Consequently, a provenance model often falls into multiple categories at the same time [RESC16].

The second dimension of this taxonomy is the purpose of a provenance model. It describes the problem or task it aims to solve, e.g. replication of prior results. Six categories can be distinguished [RESC16]:

Recall

One of the most fundamental tasks that can be accomplished with provenance is recall of past actions. The provenance information acts as a detailed log for the analyst to review. Here, the focus lies on remembering one's own activity rather than the work of someone else (e.g. a colleague). This lets the user remember why they decided for a course of action or what paths have been explored already. Recall is especially useful if an analysis is conducted over an extended period of time or if there are prolonged breaks between sessions.

Replication / reproducibility

Replication or (methods) reproducibility [GFI16] is an essential aspect of the scientific method [DC02]. It allows past discoveries to be verified independently and built upon to derive new insights. Provenance can support reproducibility similar

to recall by providing an accurate record of a conducted analysis. Apart from simply repeating an analysis step-by-step, it is also possible to explore alternative paths by changing certain aspects of a completed analysis. For example, the CLUE model (see Section 2.5.6) enables the user to return to a previous state, change a parameter, and then automatically redo the following actions [GLG+16]. This results in a new branch in the provenance graph.

Action recovery

Another basic task supported by provenance is the recovery from mistakes made by the user interacting with the analysis platform. Even without fully-fledged provenance support, most software provides basic action recovery functionality allowing the user to undo and redo actions. However, this feature is often stack-based, and thus does not take branches into account when redoing actions. A proper provenance model overcomes this limitation by exposing the tree-like structure of the action history to the user.

Collaborative communication

A more elaborate usage scenario for provenance is supporting collaborative work of multiple analysts on the same dataset. When it comes down to collaborative analysis there are two main aspects to consider [MT14]: (1) awareness of a user about the conducted analysis of their colleagues, and (2) externalization of findings and insights for other users to review. Provenance is particularly suited to achieve the aspect of awareness by recording the activity of a user. For example, simply manually sharing an analysis session including provenance information with a colleague allows them to review and extend upon the work. More sophisticated solutions even enable an automatic exchange of provenance information in real time [EKA+08].

Presentation

The most relevant use of provenance for this thesis is the presentation of facts and insights. Similarly to collaborative communication, the goal of presentation is information exchange. However, in this case the information is presented to an audience that is not necessarily involved in the analysis. Provenance information conveys how findings were discovered, and thus rationalizes the taken approach. Furthermore, provenance may be utilized in conjunction with storytelling to support the iterative fashion of the scientific method (see Section 2.5.6). That is, consumers of a story outlining a discovery may use the provided provenance information to review, adapt, and ultimately extend the conducted analysis to arrive at new conclusions.

Meta-analysis

The final use of provenance is to review and optimize the analysis process itself. This includes finding computational bottlenecks in a pipeline or origins of failure in a completed experiment [CAB+14]. Both prospective and retrospective provenance can be utilized to optimize workflows, whereas the latter is more straightforward to

use. In this case, metadata gained from completed experiments are incorporated into future runs. The other approach is to evaluate and adapt a running experiment on the fly. For example, SCIRun [PJ95] lets users model a scientific workflow consisting of modules that transform input data and pass them to other modules. During the execution of such a workflow, debugging information such as CPU and memory usage can be observed at each module, and the experiment is adapted accordingly. Besides analyzing the technical aspect of a workflow, provenance may also be used to study and evaluate the strategies and decision making of an analyst. Such evaluation is valuable to improve a user's performance or train new users [RESC16].

Just like for the different types of provenance, the purpose of a specific provenance model often belongs to several of the above categories [RESC16]. That is, provenance is usually captured and queried to achieve more than a single, simple task. Moreover, some purposes such as recall are fundamental tasks that are prerequisites for other purposes. For example, a provenance model that aims to enable replication also inherently facilitates recall. There is also an intrinsic relation between the type and purpose of provenance. In order to support a certain task or workflow, the proper kind and resolution of information has to be available. Thus, a provenance type may be more appropriate for some purposes than others. For instance, action recovery is closely related to provenance of interaction. Aside from its type, the proper resolution or granularity of provenance depends on its purpose. The granularity of provenance describes the size of the smallest primitive that can be recorded [CAB+14]. A high granularity corresponds to a detailed, fine-grained representation, while low granularity results in coarse-grained data. Recording provenance at a high level of detail results in increased storage overhead [CCM09]. Likewise, the amount of noise that has to be filtered when querying the provenance increases with its granularity [CAB+14]. Hence, the computational cost of processing that information increases accordingly [RESC16]. As a consequence, it is advantageous to capture provenance at the lowest possible granularity that is needed to complete the intended tasks. However, capturing provenance at an insufficient level of detail leads to unintended abstraction, and thus uncertainty in the metadata [RESC16]. The n-by-m problem demonstrates this by means of a process that transforms n input values to m output values [CAB+14]. Each of the m outputs may be derived from all or merely a subset of the n inputs. Now consider a provenance system that tracks the transformations performed by the process. If the granularity is too low to represent the individual computations, the provenance maps each of the n inputs to each of the m outputs. This results in inaccurate or even erroneous information when querying the recorded provenance. Therefore, it is critical to determine the appropriate level of granularity in respect to the domain as well as the intended purpose of the provenance.

3.2 Querying provenance

Regardless of its type and intended purpose, provenance is only useful if the information can be accessed in an efficient manner. Since provenance graphs in real world scenarios are to be expected to contain millions of nodes [MS11], strategies to let users extract interesting parts of the data have to be employed. There are two main paradigms for querying provenance: directed and exploratory queries [CAB+14]. Directed queries allow users to find answers to precise questions, and hence are convenient when the user knows what they are looking for. Exploratory querying lets users inspect provenance freely, usually by means of visual metaphors such as graphs. In practice, it is possible to combine both approaches: visualizing the graph gives the user a rough overview about the data; subsequent directed queries let the user extract specific and detailed information. In the following, both paradigms and examples thereof are discussed in more detail.

3.2.1 Directed queries

Directed queries are usually formulated in a query language, which may either be specifically designed for provenance or is an extension to an existing one [CAB+14]. Using extensions of general-purpose query languages such as SQL is sensible if the users are already familiar with the syntax of that language. However, such languages tend to be closely related to the underlying storage system, which can make even simple queries complicated and verbose [FKSS08]. Domain-specific languages (DSLs) overcome this limitation by making the storage details transparent to the user but require them to learn a completely new language. vtPQL is an example of a DSL for querying provenance [SKS+08]. It is part of VisTrails, a scientific workflow system that lets users build and modify visualization pipelines [FSC+06]. The provenance model captures the changes made to these pipelines as a *vistrail*, i.e. a provenance graph where each node represents a version of the workflow and an edge represents an action that transforms one version to another one. These metadata can be used to compare different versions or to return to a previous iteration of a workflow. Furthermore, pipelines can be executed while retrospective provenance is saved in an execution log. The query language vtPQL lets users query provenance at three different levels: (1) the vistrail (vt) level involves information about the evolution of the workflow, (2) the workflow (wf) level is concerned with the modules and their parameters that make up a workflow, and (3) the execution (log) level incorporates information about past executions of a pipeline. Figure 3.2 compares two versions of the same query, one using SQL and the other one using vtPQL. The vtPQL query is much more concise and readable than its SQL counterpart, as it abstracts away the details of the underlying database.

Even if using a specialized DSL such as vtPQL, more sophisticated queries can become convoluted, making creating and parsing those queries difficult for the user. Moreover, for users without any prior experience with programming or scripting languages, making directed queries in textual form can be challenging regardless of the used language. In such a case, approaches based on a GUI are more appropriate. VisTrails incorporates

```
SELECT Execution.ExecutableWorkflowId, Execution.ExecutionId,
       Event.EventId, ExecutableActivity.ExecutableActivityId
from Execution, Execution_Event, Event,
       ExecutableWorkflow_ExecutableActivity, ExecutableActivity,
       ExecutableActivity_Property_Value, Value, EventType as ET
where Execution.ExecutionId = Execution_Event.ExecutionId
and Execution_Event.EventId = Event.EventId
and ExecutableActivity.ExecutableActivityId
    = ExecutableActivity_Property_Value.ExecutableActivityId
and ExecutableActivity_Property_Value.ValueId = Value.ValueId
and Value.Value = Cast('-m 12' as binary)
and ((CONVERT(DECIMAL, Event.Timestamp) + 0) % 7) = 0
and Execution_Event.ExecutableWorkflow_ExecutableActivityId
    = ExecutableWorkflow_ExecutableActivity.
       ExecutableWorkflow_ExecutableActivityId
and ExecutableWorkflow_ExecutableActivity.ExecutableWorkflowId
    = Execution.ExecutableWorkflowId
and ExecutableWorkflow_ExecutableActivity.ExecutableActivityId
    = ExecutableActivity.ExecutableActivityId
and Event.EventTypeId = ET.EventTypeId
and ET.EventTypeName = 'Activity Start';
```

(a) SQL

```
wf{*}: x where x.module = 'AlignWarp'
          and x.parameter('model') = '12'
          and (log{x}: y where y.dayOfWeek = 'Monday')
```

(b) vtPQL

Figure 3.2: Example of a provenance query using two different languages. (a) The SQL query is verbose as the underlying storage has to be regarded, (b) the vtPQL version is much more concise as the language is specifically designed for provenance querying. The query searches for all workflows that contain a module *AlignWarp* with a parameter *model = 12* that were executed on a Monday.

Source: [FKSS08]

a feature called query-by-example [SVK+07], which lets the user search for pipelines based on a reference that is constructed visually. In particular, the reference that acts as a search criterion is built with the same interface that is used to create the pipelines in the first place. This ensures that a user who is already familiar with the main functionality of VisTrails — namely building visualization pipelines — can query provenance data without having to accustom themselves with an entirely new interface. The reference pipeline is compared to all the pipelines in the provenance graph, trying to find those pipelines that contain the reference. As pipelines are graphs themselves, this comes down to a graph matching problem that is reducible from the maximum clique problem and thus NP-complete [SVK+07]. Consequently, VisTrails employs heuristics to find matching pipelines, which yield inaccurate results in some cases. Another limitation of this implementation is that query-by-example takes only the structural properties of the pipelines into consideration. In other words, query-by-example only operates on the workflow level, whereas vtPQL allows for more expressive queries.

Stitz et al. present KnowledgePearls [SGP+19], an extension to CLUE (see Section 2.5.6) that adds functionality for searching visualization states based on their content. Each visualization state consists of metadata (e.g. title, author, creation time) and two types of properties:

1. Data-related properties describe the data themselves, independent of how they are visualized. For example, a query may include all states where a specific numerical data attribute has a certain value.

2. Visualization-related properties affect the visualization technique. This includes individual parameters, such as whether an axis uses a linear or logarithmic scale. While not implemented in the prototypes presented by the authors, the visualization type may also be a queryable property. Thus, a query may include all states that use bar charts for example.

A query consists of any number of search terms, which include a property and, depending on its type, a value it is compared against. Every state in the provenance graph is compared to the search terms, and assigned a score based on its similarity. The similarity of an individual property is computed based on its type; for example, numerical values are ranked according to their absolute difference to the reference. This allows for a fuzzy search, showing results sorted by similarity rather than only presenting results that match the query exactly. Additionally, the influence of the individual search terms on the query can be adjusted by the user. As the final score is computed as a weighted sum of the similarity scores of the search terms, the user-chosen weights determine the ranking of the results. Figure 3.3 shows the GUI of KnowledgePearls. On the left side of the interface, the provenance graph is shown, enabling quick access to past states. Existing states in the graph can be used for query-by-example [SVK+07], allowing users to find similar states. To the right, the user can build queries manually to search the graph. The search field is used to define search terms based on the properties present in the visualization.

An autocomplete feature suggests the most prevalent properties in the dataset as the user types. The frequency of each suggestion is displayed alongside an indicator that shows if the property is part of the active state. Below the search and suggestion fields, the search terms of the current query are shown, including a weighting editor to adjust their importance values. The results of the query are shown at the right side of the screen. This includes all states that feature a similarity score greater than zero in regards to the query. Since consecutive states within a session are usually similar to each other [SGP+19], the results of a query may also contain states that are almost identical to one another. Rather than showing each of these states separately, a grouping scheme is applied to reduce clutter. Consecutive states that include the same number of matching search terms are summarized as a state sequence in the results view. The sequences are sorted from highest to lowest similarity score, and can be expanded by clicking; this reveals the individual states in the sequence. Moreover, the best match of each sequence is selected as its top state, which is used as a representative for the sequence. Detailed information about the top state, such as its properties and a thumbnail, is presented to describe the corresponding sequence. A limitation of KnowledgePearls is that a query is required to be a Boolean expression of the form

$$x_1 \wedge x_2 \wedge \cdots \wedge x_n$$

where x_i for $1 \leq i \leq n$ are the search terms, which cannot contain negations. Hence, it is not possible to search for states in which a certain property is absent. Likewise, a user cannot search for states containing any number of a set of properties. In contrast to more flexible query languages like vtPQL, a query in KnowledgePearls cannot take the structure of the graph into consideration. The upside is that queries in KnowledgePearls are less convoluted, and easier to build due to the provided GUI.

3.2.2 Exploratory queries

Directed queries are not an appropriate means to gain insights, if the user does not exactly know what they are looking for a priori. Instead, the user has to be presented with an overview of the provenance data, allowing them to explore aspects that they deem interesting in more detail. Usually, exploratory querying makes use of visual representations to take advantage of the human capability to detect patterns that way [CAB+14]. Yet, the data have to be abstracted or filtered before they are presented to the user, as the volume of provenance can become overwhelming. Node-link diagrams, which are the most common way to visualize provenance data [BYB+13], are particularly susceptible to this problem, since they do not scale well beyond hundreds or even thousands of nodes [BYB+13, MS11]. Naively rendering such a high number of nodes at once, makes analyzing and navigating the graph difficult for the user.

3.2.2.1 User views and hierarchical grouping

One way to overcome the complexity of large datasets, is letting the user declare what parts of the data are relevant, and subsequently derive simplified representations called

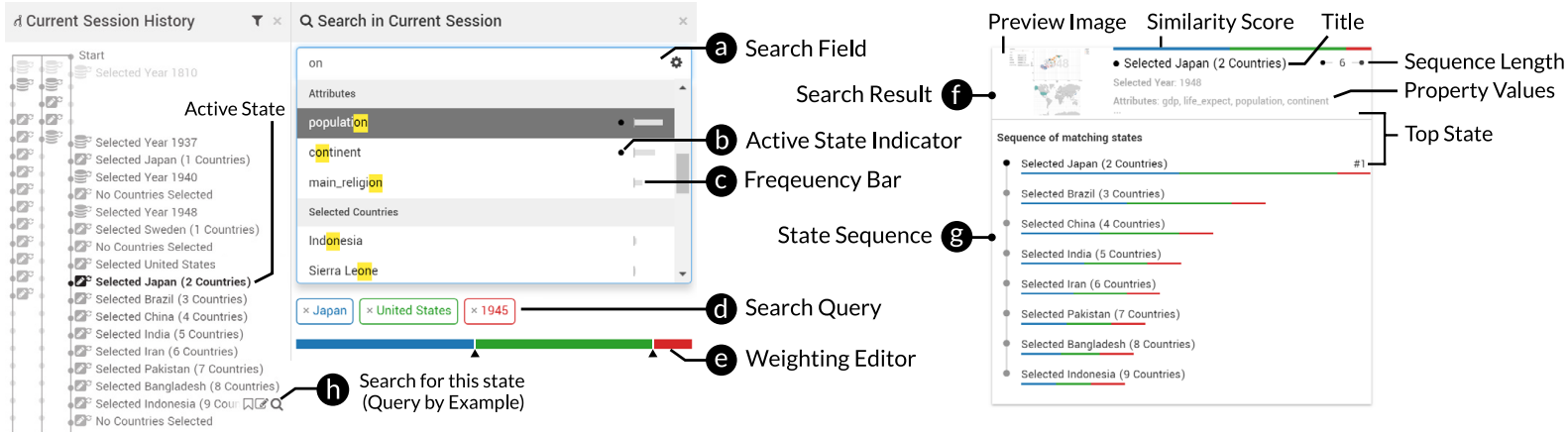


Figure 3.3: GUI of KnowledgePearls. A search field is used to input property names (a), suggestions are shown with indicators for properties of the active state (b) and their frequency in the data (c). The weights of the search terms in the query (d) can be adjusted using a simple editor (e). The search results (f) are presented as a list of state sequences (g), grouping together states that are similar to one another. Existing states in the graph can be used for query-by-example (h).

Source: Stitz et al. [SGP+19]

user views [BCDH08]. User views hide irrelevant dependencies (i.e. edges) between nodes by grouping those nodes together as a single node in the simplified graph. Given a provenance graph and a set of relevant nodes thereof, a user view should (1) contain a group for each relevant node, (2) preserve the dependencies between relevant nodes, and (3) minimize the number of groups in the final graph [BCDH08]. Biton et al. present a grouping algorithm with an asymptotic complexity of $\mathcal{O}\left(|N|^2 + |E|\right)$ where N and E are the input sets of nodes and edges respectively [BCDH08]. While this algorithm operates in polynomial time and guarantees requirements (1) and (2), the computed solution is not minimal in all cases. A fundamental advantage of user views is their flexibility; an abstraction of the provenance graph may be suited for one user but inadequate for another one, since different users may have varying tasks and intentions. User views accommodate for this variety by letting the users define which of the nodes are relevant to them. This results in multiple user views of the same data, each of which is useful for a different array of purposes.

The Provenance Map Orbiter [MS11] employs a hierarchical grouping strategy similar to user views. It is designed primarily with filesystem provenance in mind, but can be used to visualize any type of provenance graph. Like with simple user views, certain nodes are declared as relevant by the user, which controls the applied grouping scheme. In contrast to user views, the resulting grouping is multilayered; groups can contain other groups, yielding a graph with multiple layers of abstraction. These layers can be explored interactively by the user via semantic zoom [BH94]. The visualization starts with the highest level of abstraction giving the user a broad overview of the data. As the user zooms into the graph, the contents of the groups are revealed gradually, presenting more and more detail. This lets users explore the provenance graph at highest detail without overwhelming them when they first start their analysis. Another benefit of this hierarchical approach is its reduced computational cost relative to a naive method. The graph is not rendered in its entirety at full detail, making the graph layout algorithm more efficient as not all nodes have to be regarded.

InProv [BYB+13] is another tool intended for exploring large filesystem provenance datasets. At its core is a hierarchical grouping algorithm as in the previous example; however, the provenance graph is not visualized as a traditional node-link diagram in this case. Instead, InProv represents the graph using a radial layout. Nodes are aligned as segments of a ring with edges connecting the nodes inside of it. Figure 3.4 shows a screenshot of the GUI as well as a schematic depiction of its elements. The radial layout was chosen because it emphasizes the semantic rather than the spatial relations between nodes [BYB+13]. In order to better manage the large size of provenance data, a grouping algorithm based on timestamps is applied. Filesystem provenance is a special case of data provenance, recording the activity and dependencies of processes, files, and pipes within a system (e.g. a process writes to a file). Usually, system activity occurs in bursts separated temporally by longer periods of inactivity between them [BYB+13]. Based on their timestamps, InProv attempts to group the events of those bursts together, producing a radial plot for each of those groups. This initial grouping can still yield graphs with

a node count too high for users to easily comprehend all at once. Accordingly, nodes within a timestamp-based group can be further summarized in subgroups if their count exceeds a certain threshold. Subgroups in the radial plot are drawn as segments that are thicker than segments corresponding to single nodes. A subgroup can be expanded and explored by clicking on it, revealing its contents as a separate radial plot. Aside from the radial plots, InProv provides several GUI elements for navigation as seen in Figure 3.4. At the bottom of the screen a timeline indicates the temporal position of the currently shown plot. Here, other groups corresponding to different timestamps may be selected and explored. On the right side of the interface, a history of previous views helps users navigate to past states and keep an overview of their analysis. Similarly, the node stack in the top left corner gives a summary of the subgroup hierarchy. On top of facilitating orientation, the node stack can be used to return to a higher level in the grouping hierarchy. Finally, the temporal grouping scheme can be disabled on the fly by pressing the button in the top right corner; in its place, an alternative algorithm is used, grouping nodes together only based on their relations with each other.

3.2.2.2 Motif-based aggregation

Motif-based aggregation is yet another method to reduce the visual complexity of large graphs for node-link diagrams [MSOI+02, MRS+13]. The idea is to detect common topological patterns — called motifs — within the graph, and replace instances thereof with descriptive glyphs. A major challenge of this approach is generating motifs and finding their occurrences, as this is a subgraph matching problem and thus NP-complete [MRS+13] (see Section 3.2.1). Maguire et al. employ motif-based aggregation to visualize large graphs representing workflows of biological experiments [MRS+13]. Motifs are extracted from the workflow by considering both the topology of the graph as well as the semantics of its nodes. The extracted motifs act as candidates for a subsequent semi-automatic selection process. The candidates are sorted according to various indicators, such as their frequency in the workflow, and presented to the user. The final choice of which motifs are applied is left to the user, since choosing the optimal motif requires detailed domain knowledge [MRS+13]. When the motifs are applied, the matching subgraphs are replaced by glyphs, which are rendered at three different resolutions that are changed via semantic zoom [BH94]. The lowest resolution of a glyph only gives a rough overview about the topology and node types of the corresponding subgraph. As the user zooms into a glyph, its representation changes to versions with higher resolution. At the highest level of detail the original subgraph with all of its details is revealed. Adaptive Visualization of Comprehensive Analytical Data Origins (AVOCADO) is a platform for visualizing provenance graphs of analyses in the context of biomedical research [SLSG16]. AVOCADO combines three approaches to reduce the visual complexity of the provenance graph: (1) hierarchical aggregation groups together nodes based on the inherent semantics of the dataset, (2) motif-based aggregation simplifies those groups further based on used workflow templates, and (3) a DoI function dynamically expands and collapses nodes based on attributes, such as if the node is part of an active filter or if it is selected by the user.

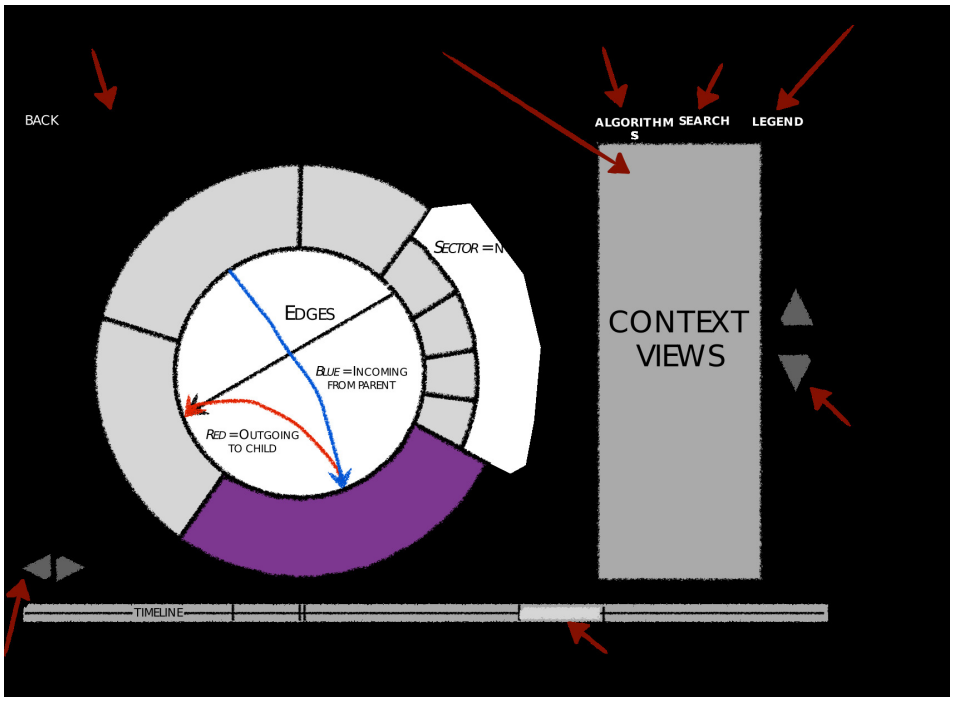


Figure 3.4: InProv uses a radial layout for visualizing graphs of filesystem provenance in a hierarchical manner. Nodes are grouped according to their associated timestamp so that nodes within a group are close to each other temporally. Nodes of a group are presented as segments of a ring; the group to be visualized can be selected via a timeline interface at the bottom of the screen.

Source: Borkin et al. [BYB+13]

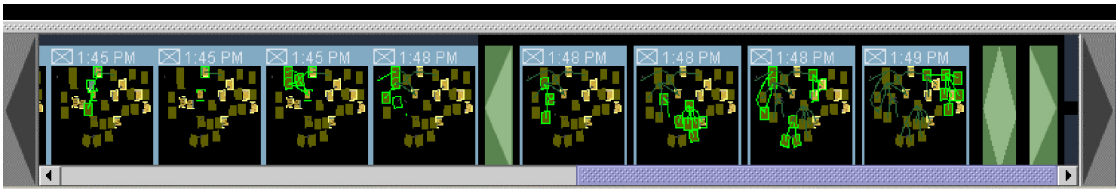


Figure 3.5: Inline provenance tree as used in Designers' Outpost. Nodes correspond to session states and are represented by thumbnails with a timestamp. Branches can be collapsed and expanded in place. Diamonds correspond to collapsed branches; expanding a branch divides the diamond in half, showing the content in between.

Source: Klemmer et al. [KTPG+02]

3.2.2.3 Inline layouts

While node-link diagrams lend themselves to depict provenance graphs in a natural manner, they require a lot of screen space [HMSA08]. Inline designs are a space saving alternative to visualize acyclic, connected graphs (i.e. trees). The nodes are arranged in a single line showing all branches one after another. Klemmer at al. choose such a layout for visualizing the history of a session in Designers' Outpost [KNF+01], a solution for collaborative web design that combines aspects of digital and analog sketching on a blackboard [KTPG+02]. The users can put handwritten sticky notes on a large screen that is monitored with two cameras. Notes are detected and stored in digital form, enabling users to view and manipulate notes from past sessions even if their physical counterparts are not available anymore. Notes can be moved around, put into groups, and connected via digital lines drawn on the screen. Affinity diagrams can be created, a common tool for brainstorming and exploring ideas [AC12]. Provenance is captured and shown at the bottom of a screen as seen in Figure 3.5. The nodes of the provenance tree correspond to states in the session and are shown as thumbnails with a timestamp. A thumbnail summarizes its state and additionally highlights the difference between it and the previous one. Branches are drawn one after another; however, each branch is collapsed by default, hiding its content. Glyphs with a diamond shape denote a collapsed branch. Clicking on the glyph splits it into two triangles, revealing the content of the branch between those triangles. Although, an inline layout of the provenance tree requires significantly less space than node-link diagrams, a user study conducted by Klemmer et al. showed that presenting multiple branches that way can be confusing [KTPG+02]. Heer et al. showcase a graphical history tool that captures visualization provenance for visual analysis [HMSA08]. Their prototype uses a similar inline layout to visualize the provenance tree.

CHAPTER 4

Design space analysis

There are a lot of design options to consider when developing a provenance-based storytelling mechanism for PRo3D, given the vast amount of existing work on storytelling and provenance discussed in the two previous chapters. In order to find a solution that builds upon the most relevant aspects of existing work, it is necessary to identify those aspects first. To this end, we (1) analyzed the workflow of Digital Outcrop Model (DOM) interpretation in PRo3D, (2) used actual geological story presentations based on PRo3D analyses as a reference, and (3) collected domain expert feedback. As a result, we identified the following five key requirements for an integrated storytelling solution in PRo3D:

R1 Stories need to cover multiple locations and outcrops at multiple scales, while conveying the spatial context during presentation.

R2 Stories have to be interactive, in particular two modes of interaction need to be supported:

- **R2.1** Presentations have to support user interaction to view the data from different angles on the fly.
- **R2.2** Story playback needs to be paused at arbitrary points to switch to a fully interactive analysis mode.

R3 Users must be able to augment stories with labels, which describe either a scene as a whole or specific locations in the dataset.

R4 Stories must be able to incorporate all relevant stages of an analysis process.

R5 The storytelling features are required to integrate with the existing DOM interpretation workflow in PRo3D rather than modifying it substantially.

These requirements act as a basis for our design decisions, which we discuss in more detail in later sections of this chapter. The remaining part of this section explains the reasoning of how these requirements were derived.

Spatial context As **R1** declares, an essential aspect of geological stories in PRo3D is being able to view the data from different locations and at different scales. This is an immediate consequence of the DOM interpretation workflow, which is illustrated in Figure 4.1. Geoscientists analyze outcrops in a hierarchical manner: First, they measure the dimensions of the outcrop itself. Subsequently, the main geological units are identified, which segment the outcrop according to properties such as color, weathering characteristics, and bed style. For each unit, measurements are taken and contained sedimentary structures and properties are analyzed. Finally, the results are summarized in graphical logs such as sedimentary logs, which visualize beds and sedimentary rocks using simple patterns and symbols [Nic09]. This process is repeated for multiple locations and outcrops to arrive at a large-scale model of the investigated area [HGE+11, HGS+11]. The structures and properties that are measured during this process vary greatly in scale, and studied outcrops may be located hundreds of meters apart from each other. For example, Victoria on Mars is an approximately 75 m deep crater with a diameter of about 750 m that contains multiple outcrops, which were imaged during the MER mission [HGE+11]. The outcrops have a vertical and lateral extent of up to 12 m and 25 m respectively [HGE+11]. However, individual beds of these outcrops are as thin as a few centimeters [HGE+11], and can generally be even at millimeter-scale [Nic09]. Stories that aim to convey the whole geoscientific analysis process are required to show the data at these varying scales and locations, while maintaining the spatial relations for the audience. Otherwise the viewers of the story may have difficulties relating key data points to one another. As discussed in Section 1.2, this is especially difficult for traditional slide shows based on screenshots, and only possible with considerable effort from the story editor. Consequently, the first formal requirement **R1** is that our proposed solution must address this issue by clearly communicating spatial relationships to the audience.

Interaction The second requirement **R2** consists of two parts and is concerned with user interaction during the presentation of a story. As discussed in Section 2.3 interactive storytelling is an effective way to increase the comprehensibility and credibility of a narrative, while also being more engaging for the audience [MLF+12]. These benefits are mostly relevant for self-running presentation scenarios, where the viewer has an opportunity to explore the data on their own. Still, other scenarios in which a presenter is responsible for delivering the story (e.g. live talks and dynamic discussions) can benefit from interaction as well. For instance, interaction allows presenters to adjust the story on the fly, diverting from the original path that was laid out by the scripter. This is necessary when questions from the audience cannot be answered using the portions of the data that were prepared a priori. Static artifacts like screenshots and videos, as used in traditional slide shows, are impractical for such dynamic situations; an integrated storytelling solution presents the original data in any case, and thus should be able

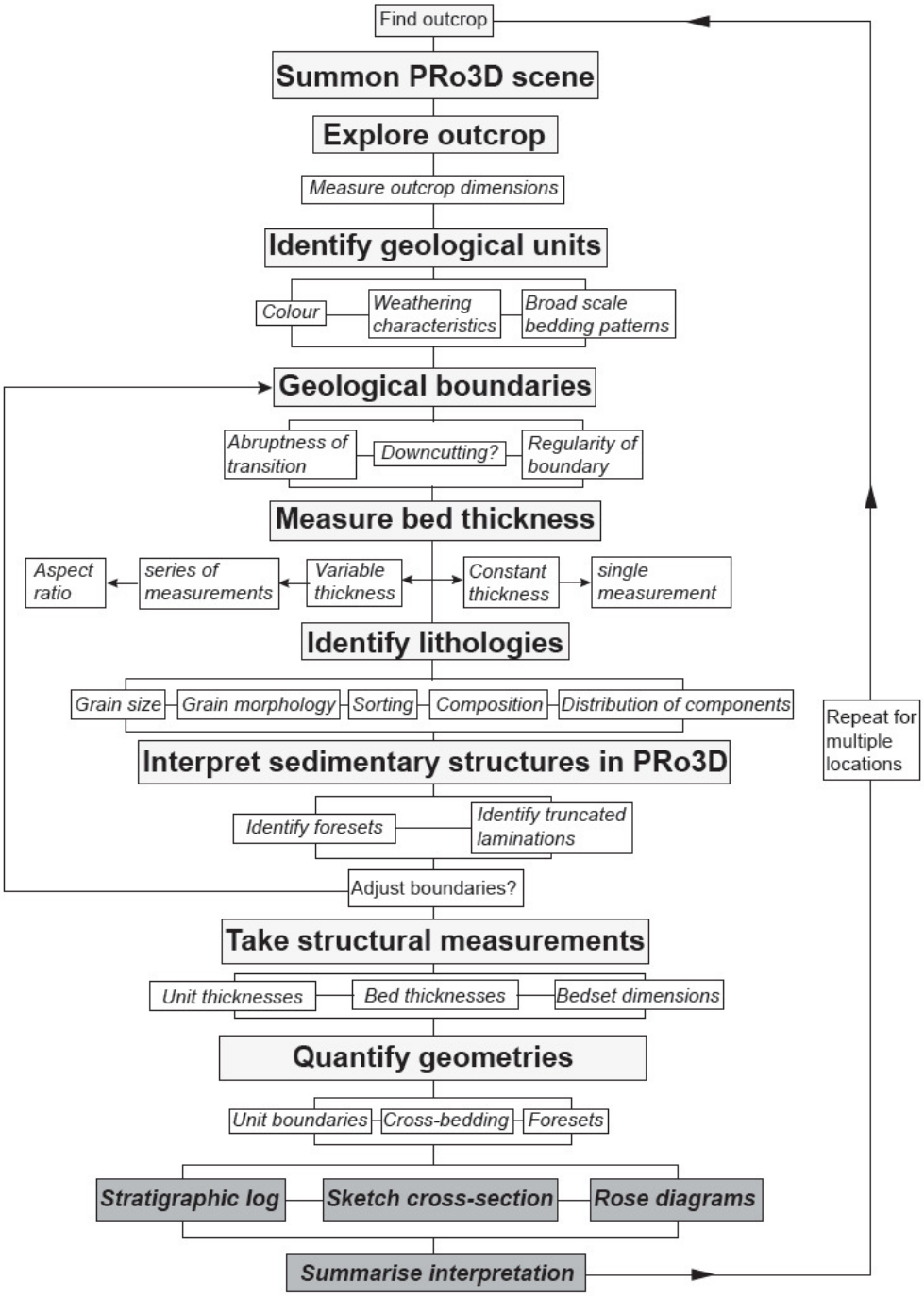


Figure 4.1: Interpretation of DOMs in PRo3D.

Source: Barnes et al. [BGT+18]

to provide means of user interaction easily. Even a minimal level of interaction — i.e. viewing interaction — improves the understanding of 3D spatial relations during the presentation [WH07], making meeting **R1** more reliable. Therefore, **R2.1** requires our solution to support viewing interactions during a presentation. **R2.2** refers to the possibility of provenance-based solutions to inherently enable higher levels and more comprehensive types of interaction. Linking provenance information to stories allows users to switch back to the analysis stage at any time, exposing the entire interactive feature set of the analysis platform. A user may take advantage of this by pausing the story playback at certain points to explore aspects not covered by the story or verify specific claims, and then return to the original narrative. Alternatively, a martini glass structure may be employed to build upon a story by using it as a starting point for studying a new hypothesis. Both verifying and building upon previous work are essential aspects of the scientific method [DC02, Ple18], which can be achieved by satisfying **R2.2**.

Text labels According to **R3** there has to be an option for users to add textual annotations to stories. We devised this requirement primarily by evaluating existing slide show presentations demonstrating the DOM interpretation capabilities of PRo3D. These stories were created using tools such as PowerPoint, but can serve as a reference to identify common narrative structures and aids used in geological stories. Figures 4.2 and 4.3 show two slides of such a presentation; the data originate from the Hanksville-Burpee Dinosaur Quarry (HBDQ) near Hanksville, Utah, USA. The slides consist of a cropped screenshot of the DOM visualized in PRo3D, and additional textual annotations describing the scene. In particular, two types of annotations can be distinguished in the first slide: (1) a slide title describing the scene as a whole, and (2) labels anchored in the 3D scene pointing at specific locations. In an effort to support the creation of similar geological stories in PRo3D, **R3** requires our solution to provide means to add such annotations to the story. An interesting observation to be made is that Wohlfart and Hauser identify the same basic types of annotations for their narrative volume visualization solution of medical data [WH07]. This suggests that despite the very specific application domain of planetary geological analysis, deliberations in this context may generalize to 3D storytelling problems in other domains. Additionally, the second slide in Figure 4.3 includes annotations specific to the geoscientific domain. At the left side of the screenshot a double arrow is used to describe the height of the outcrop alongside a label displaying the exact length in meters. This special form of anchored annotation is frequently used in geological stories to depict the extent of various features of outcrops. To the right of the screenshot, a summary log has been added, showing sedimentary structures, grain sizes, and paleocurrent directions. At the time of writing, PRo3D does not support the generation of such diagrams. As a consequence, an integrated storytelling solution in PRo3D cannot provide means to add such illustrations at this point, and was omitted from **R3** for the scope of this thesis.

Stages of interpretation Satisfying **R4** requires that stories do not only present individual findings, but also the various stages of the analysis process that lead to those

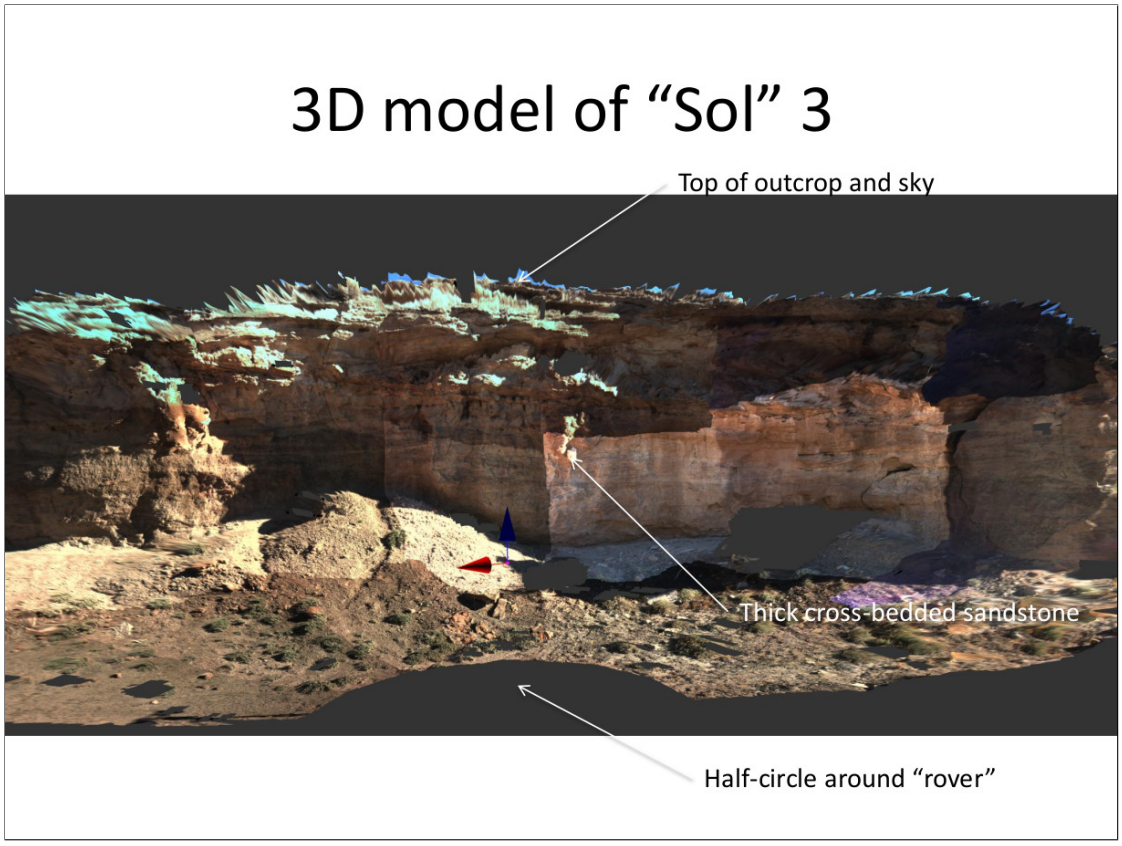


Figure 4.2: Slide of a PowerPoint presentation showcasing the DOM interpretation possibilities in PRo3D. Two basic kinds of annotations were added manually: (1) a header describing the scene as a whole, and (2) textual descriptions of specific points.

Source: Presentation by Marijn van Cappelle 2016

results. This facilitates the reproduction and verification of results, as it is difficult to verify a finished model without any information about the underlying reasoning process [LHV12]. Moreover, verification and confirmation of results is essential when working with incomplete or low-quality data, since geological models are highly variable and depend on the experience of the geoscientist in such cases [BGSJ07]. Geological analyses of the Martian surface are susceptible to this kind of *conceptual uncertainty* [BGSJ07]; direct field observations are impossible, making the interpretation process dependent on DOMs derived from rover and orbiter imagery of varying resolution. The human bias inherent to interpretations of such imperfect data has to be addressed by conveying the analytical process as a whole. The DOM interpretation workflow, as described in Figure 4.1, lends itself to be divided into different stages. Its hierarchical structure allows the interpretation of an outcrop to be regarded as a series of dependent milestones, which can give insight into the rationale of the analyst. For example, such milestones

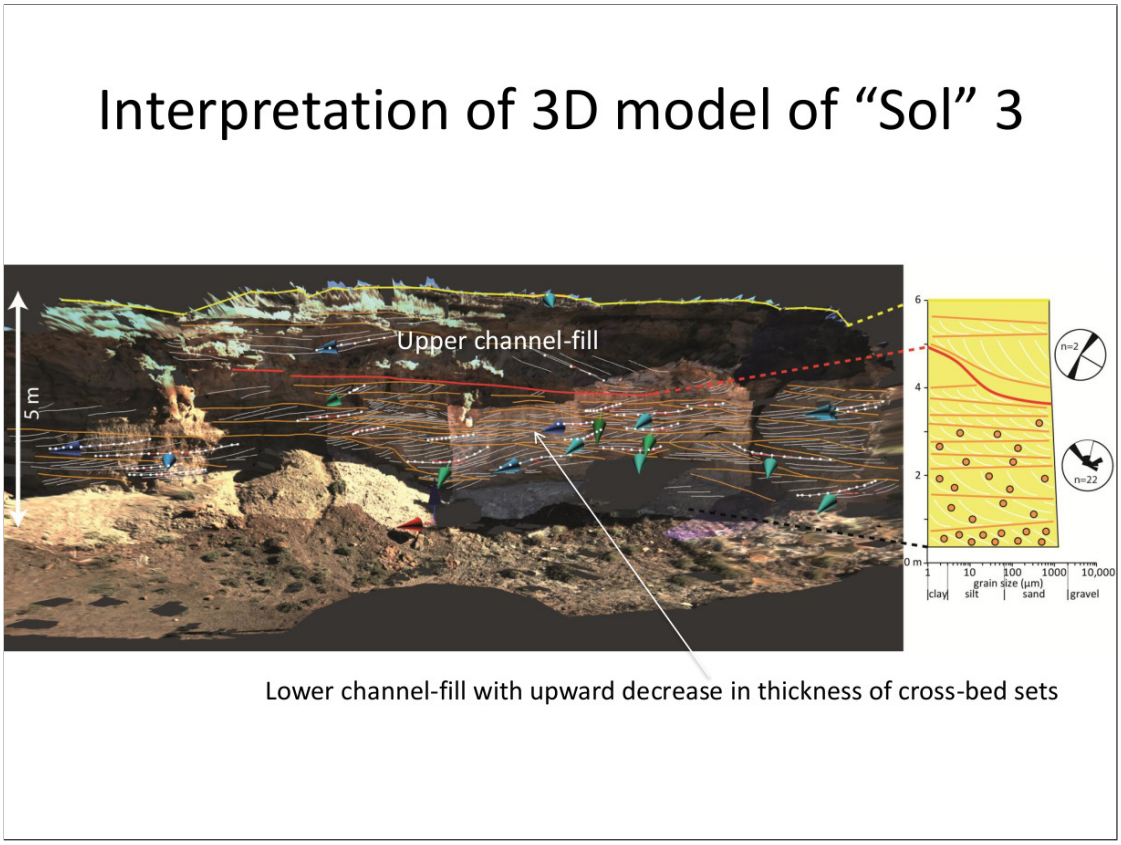


Figure 4.3: Another slide from the same presentation as the slide shown in Figure 4.2. A double arrow is used to depict the vertical extent of the outcrop, and a sedimentary log created outside of PRo3D summarizes various features in a simplified manner.

Source: Presentation by Marijn van Cappelle 2016

may include the identification of main geological units, measurements of individual beds, and the interpretation of finer sedimentary structures. Ideally, a provenance system can automatically capture the different stages of an analysis, allowing the editor to easily access and incorporate them in a story.

Integration The final requirement **R5** is concerned with the integration of the storytelling features into the existing framework of PRo3D. At the time of writing, PRo3D is still being actively developed, and functionality may be changed or added as a consequence. Likewise, the analysis workflow itself may evolve as features are added. To accommodate for eventual changes, **R5** requires the integration of our storytelling solution to be as independent from this workflow as possible. This also means that the storytelling mechanisms cannot change the existing workflow substantially. For example, the provenance system must not require user intervention to an extent that interferes

with the analysis process itself. Rather, provenance is to be captured automatically and transparent to the user.

In the remainder of this chapter we explore the design space of storytelling with respect to PRo3D in more detail, using the defined requirements to identify the most reasonable options. These deliberations are mostly of theoretical nature giving an overview of various design choices; yet, we also mention what approaches we chose for our solution, which is presented in Chapter 5. First, we discuss how a story in PRo3D may look like and what elements it contains. Then, we examine how and what kind of provenance needs to be captured, and how it may be presented to the user in a comprehensible manner. Finally, we explore the role of the camera in the 3D scene and how it may be handled in the context of our provenance-based storytelling system.

4.1 Story form

A fundamental design choice is the form of representation that is used for visual data stories within PRo3D. In Section 2.1, seven different story genres by Segel and Heer [SH10] were discussed. The authors identified these genres by surveying a large body of visualizations, mostly in the context of online newspapers. The studied examples deal with 2D data for the most part, while storytelling in PRo3D has to process and present full 3D datasets. As a consequence, these existing solutions are only partially applicable to the use case at hand. As established by **R1**, conveying spatial relations to the audience is a central concern. Out of the seven genres, animations are inherently suited best to address this issue. Animation of the viewpoint helps viewers build a mental map of the shown data, facilitating understanding and tasks like reconstructing spatial structures [BB99]. The animation story genre according to Segel and Heer includes only non-interactive animations, also called data videos [AHRL+15]. While a popular form of storytelling [AHRL+15], data videos do not offer any interaction (contradicting **R2**) and are primarily intended for self-running presentation scenarios targeting a broad audience. Nevertheless, animation can also be incorporated in interactive story forms by combining multiple genres. A common approach is to let users build their stories by defining visualization states as keyframes of the animation [WH07, AWM10, LHV12, GLG+16], and handle the story progression similar to a slide show. The animation is the result of interpolating the visualization states that were defined as keyframes. To this end, we consider a visualization state to consist of two distinct components:

View artifacts
Properties that are mostly irrelevant for the state of the analysis and only affect the visualization itself are considered view artifacts. They are not tracked in the provenance graph, and are only relevant for story keyframes. In the case of PRo3D, view artifacts include:

- general rendering parameters, e.g. the size of dip and strike planes and arrows as seen in Figure 1.2.

- rendering parameters of individual surfaces in the DOM, such as their tessellation quality and texture to apply.

The current state or value of a view artifact does not provide meaningful information about the analysis stage. For example, the rendering quality of a surface does not give insight about the interpretation progress or the rationale of the analyst at the time corresponding to that visualization state.

Change artifacts

Contrary to view artifacts, change artifacts are properties that describe the state of the analysis. Whenever such an artifact is modified, it triggers the provenance system and the changes are applied to the provenance graph, for example by appending a new node to the graph. Note that change artifacts can also include parameters that affect the rendering or visualization itself. For example, the color of a polyline is a change artifact. The difference to other visual properties that are classified as view artifacts is that the line color carries a semantic meaning relevant for the analysis. Changing this color implicitly changes the semantics of the line, and thus modifies the state of the interpretation. We identify the following change artifacts:

- visibility and depth ordering priority of surfaces.
- geoscientific annotations and their properties (e.g. visibility, color, line thickness, text)

The DOM interpretation process revolves around identifying, measuring, and annotating features of an outcrop. Accordingly, change artifacts in PRo3D primarily involve annotations made throughout the interpretation. Additionally, the visibility and depth ordering of surfaces are adjusted during the analysis, depending on which surface is being interpreted. Changes made to these artifacts are processed and stored by the provenance system, which is discussed in Section 4.2. A story keyframe does not save change artifacts directly, but simply references the corresponding node of the provenance graph.

The camera position and orientation do not necessarily fall into either of these categories. As discussed earlier, the DOM workflow includes the identification of features at highly varying scales and locations. Likewise, the camera is adjusted as the analysis progresses, linking the camera to the analysis state to some degree. However, often small camera movements are made to adjust the view during the analysis without affecting its state. In these cases, it would be suboptimal to capture the camera changes in the provenance graph. We examine possible ways to handle the ambiguity of camera-related properties in Section 4.3 in more detail.

To obtain a smooth transition between two visualization states, the values of their artifacts have to be interpolated. Depending on the type, different transitions are applied. For example, using the taxonomy by Heer and Robertson [HR07], at least viewpoint,

filtering, and visualization transitions are required to model changes to the visualization states. Viewpoint transitions aid in retaining spatial relationships (**R1**) [BB99], and are achieved by interpolating the camera-related properties. The position can be interpolated linearly; for the orientation (represented as a quaternion) spherical linear interpolation can be applied to accomplish a constant-speed transition [Par12]. Filtering transitions are used to model changes such as appearing or disappearing lines (e.g. by alpha blending), visualization transitions are concerned with visual properties such as color and thickness of lines. While the two latter types of animations are not necessary according to our requirements, it is shown that such animations can be applied to improve the graphical perception of changes for users [HR07].

Going beyond a simple keyframe animation, stories in PRo3D can also be combined with the slide show genre by regarding the keyframes as slides. In the following, we refer to these slides as frame slides. Users build stories using a storyboard interface as commonly seen in applications such as Microsoft's PowerPoint, making the familiarization with the interface easier. The story progress can either be automatic (e.g. CLUE), or controlled like a traditional slide show with keyboard input. In the latter case, viewing interaction, as required by **R2.1**, can be realized while viewing an individual slide. Gratzl et al. build upon this concept and introduce a second type of slide that is used for text captions [GLG+16]. For PRo3D, this idea can be adapted to allow slides with content independent of the outcrop data. These slides may look similar to traditional slides (i.e. formatted text, bullet lists, images) and be used to introduce general information about a location or the mission the data originate from. Frame slides and these text slides can be combined arbitrarily in a story, whereas animated transitions occur between two frame slides. To satisfy **R3**, text annotations may be added to frame slides, describing the visualization in more detail similar to an annotated chart. According to this design, a story in PRo3D is a combination of animation, traditional slide show, and annotated chart with support for presenting original data with user interaction.

4.2 Provenance

A provenance-based storytelling design is inherently suited to integrate different steps of the analysis (**R4**) and allows for arbitrary transitions between the exploration and presentation stages (**R2.2**). The capability to switch back to the exploration stage at any point during the presentation, enables reproducibility and the iterative workflow of the scientific method as exemplified by CLUE (see Section 2.5.6). The core of such a design is the provenance system itself; it has to be implemented in a manner that it may operate without user intervention to preserve the original analysis workflow in PRo3D (**R5**), and also capture all the essential steps of the interpretation process (**R4**). In the remainder of this section, we discuss crucial design choices that allow a provenance system to support storytelling in PRo3D while meeting these requirements.

4.2.1 Type and representation

A fundamental design decision is concerned with the type and granularity of provenance that is captured. As discussed in Section 3.1, the proper choice of these characteristics inherently depends on the intended purpose of the provenance system. The main purpose of provenance in PRo3D is storytelling (*presentation*). To support this task, past states of the analysis have to be accessible (*recall*) and presented findings have to be reproducible in the context of a scientific workflow (*replication / reproducibility*). Ideally, provenance should be represented at a level of abstraction that resembles the individual stages of the DOM interpretation workflow seen in Figure 4.1. Directly tracking all change artifacts (*data provenance*) and user actions that affect them (*interaction provenance*) results in a large graph with unnecessary detail. For example, suppose a single point in a polyline annotation is slightly moved; whether this change should be reflected in the provenance graph depends on its semantics in the context of the analysis. If it does not affect the state of the interpretation — i.e. it does not constitute a new finding — it may be ignored for our purposes as it would not add any information to a resulting story. However, automatically deciding on the significance of such a change (*insight provenance*) based on a fixed set of rules is difficult, since PRo3D does not store any meta information about the interpretation process and annotations do not carry any explicit meaning. A manual approach (e.g. the user decides when a new milestone is reached) may be possible [RESC16], but violates the requirement for an unintrusive solution (**R5**).

As a consequence, a solution operating at a low level of abstraction is the best option to meet our requirements at this point. User interactions are observed and values of the change artifacts (as described in Section 4.1) are tracked to identify events that are potentially relevant for the analysis process. For PRo3D, the following events can be identified:

Add annotation
The main feature of PRo3D is the ability to add annotations to the DOM. Whenever a new annotation is added, the state of the interpretation is likely to have changed. Every such event has to be captured in the provenance graph, as to not skip a potentially crucial interpretation step (**R4**).

Edit annotation
Existing annotations can be modified by adjusting their properties including 3D positions, line color and thickness, and visibility state.

Delete annotation
Annotations can also be removed, representing the inverse operation to the addition of an annotation.

Load annotations
At the time of writing, PRo3D allows users to export and import annotations from disk. Importing annotations is basically the same as adding annotations, but

handled separately for easier understanding of the provenance graph. In the future, the import and export feature of annotations may be expanded to handle whole analysis sessions (including provenance) making this event type obsolete.

Modify groupings
Annotations can be grouped together in a recursive, tree-like fashion similar to a filesystem hierarchy: a group can contain annotations and other groups. The grouping does not carry any explicit semantics and is chosen at the discretion of the user. Usually, annotations are grouped according to outcrops and further divided by type, e.g. unit boundaries, bed sets, and cross-beddings. Like annotations themselves, their grouping implicitly represents the state of the analysis, when adhering to such a scheme. As the grouping changes, the state of the interpretation likely changes as well, making such events relevant for provenance.

Edit surface parameters
DOMs interpreted within PRo3D can consist of multiple surfaces originating from different data sources. For a single location multiple surfaces may overlap, requiring the user to compare and inspect multiple data sources. To this end, the user may modify the visibility and depth ordering of the surfaces. As the interpretation process progresses, these surface parameters are adjusted according to the location that is being inspected. Events related to these parameters are significant for provenance, since their state is tied to the interpretation stage.

A node in the provenance graph corresponds to a state of the change artifacts, while an edge represents an event that leads from one state to another one. We opt to represent provenance as a directed rooted tree, that is an acyclic connected graph $G = (V, E)$ with a root $r \in V$ for which there exists exactly one directed path to every other vertex $v \in V$ (also called *arborescence*) [Gal17]. The root r represents the initial (empty) state of the analysis. In practice, an analysis may contain multiple paths that lead from the initial state to any given state, rendering a tree inadequate for these situations. However, trees are easier to visualize than general directed graphs [TKE12], and our simplification strategies (see Section 4.2.2) only work for arborescences. We argue that the situations that cannot be represented in this way are not common enough to warrant the increased complexity of processing a general graph.

Some aspects of the program state in PRo3D are not tracked, since provenance is limited to change artifacts. For example, view artifacts and user interactions with the graph itself (*provenance of provenance*) are disregarded, since they are not relevant for storytelling. Restricting provenance in that way, reduces the complexity of the resulting graph, but tasks going beyond presentation may require more information. Full action recovery is not possible with our solution, and would require a separate, more fine-grained provenance system or a hierarchical approach. However, this issue is irrelevant for storytelling in PRo3D, and thus goes beyond the scope of this thesis.

4.2.2 Simplification rules

Low-level types like interaction provenance captured at a high granularity increase the storage overhead and require querying strategies that filter out noise [CCM09, CAB+14]. To address this issue, we propose a set of simplification rules that aim to reduce the complexity of the provenance graph without removing information that is potentially relevant for a story. For the following, a provenance graph is $G = (V, E, c, s, t)$ with root $r \in V$, where

V	: set of nodes
E	: set of edges
$c \in V$	: node corresponding to the current location in the provenance graph
$s\colon V \to S$	: labeling function assigning a state to each node, with S being the set of all possible analysis states representable by the provenance system
$t\colon E \to T$	: labeling function assigning an event type to each edge, with T being the set of different event types described above

The function $succ(n) = \{ x \mid (n, x) \in E \}$ returns the children of a given node, and $pred(n) = \{ x \mid (x, n) \in E \}$ returns the parent (or an empty set iff $n = r$). Whenever an event of type $e \in T$ is triggered, resulting in a new state $q \in S$ with $q \neq s(c)$, the current provenance graph G_i has to be updated to arrive at a new graph G_{i+1} that incorporates q and e, while minimizing the number of nodes in G_{i+1}. To this end, we propose the following rules:

(1) If the current node c_i has a neighbor $n \in pred(c_i) \cup succ(c_i)$ that is identical to the new state q, thus it satisfies

$$s(n) = q$$

then it is not necessary to append a new node to the graph. Instead, the current node is changed to that neighbor resulting in $G_{i+1} = (V_i, E_i, n, s_i, t_i)$. In other words, if a state is reached that was already visited and stored in the provenance graph before, it is redundant to create a new node as the existing node can be selected as the new current one. This rule can be extended to verify that the existing edge matches the event

$$t_i(c_i, n) = e \vee t_i(n, c_i) = e^{-1}$$

where e^{-1} is the inverse event type of e (e.g. the inverse of adding an annotation is removing it again). Disregarding the event type can lead to an inaccurate graph if a transition between two states can happen due to different event types (e.g. adding an annotation and loading a single annotation). Note, that it is sufficient to only check neighbors of the current node (i.e. its parent and children) when applying this rule, since the provenance graph is a tree: Consider a candidate node $v \neq c_i$ that is not a neighbor of c_i, i.e.

$$v \notin pred(c_i) \cup succ(c_i) \wedge s(v) = q$$

There is a single path in G_i from the root r to c_i and v respectively. Simplifying the subsequent graph G_{i+1} like above would require the addition of another edge (c_i, v) with $t_{i+1}(c_i, v) = e$, which would create another path from the root to v and thus resulting in an (undirected) cycle. Consequently, only direct neighbors of c_i have to be considered for this rule, when working with arborescences.

(2) If the current node c_i is not the root r and does not have children, thus

$$pred(c_i) = \{p\} \wedge succ(c_i) = \emptyset$$

and the current event e is of the same type as the one from which c_i was derived

$$t_i(p, c_i) = e$$

then it may be possible to coalesce these consecutive events into a single one. The idea is that the intermediate results of multiple identical actions are likely to be irrelevant for storytelling, and thus can be merged into a single event. To this end, the modified graph is $G_{i+1} = (V_i, E_i, c_i, s_{i+1}, t_i)$ with

$$s_{i+1}(x) = \begin{cases} q & \text{if } x = c_i \\ s_i(x) & \text{otherwise} \end{cases}$$

In other words, the state of the current node c_i is simply replaced by the new state q. In addition to the conditions above, the applicability of this rule depends on the event type itself. Consider a sequence of modify events; if they affect the very same annotation or surface, it may be assumed that the intermediate results are not relevant and can be coalesced. Modifications on different objects, however, may qualify as independent steps whose intermediate results are potentially significant for a story. Likewise, assumptions about the importance of single annotation additions are difficult to make, due to the lack of meta information about the analysis process in PRo3D. These events should be exempted from this simplification rule. Consecutive removals of annotations are usually steps of an undo action and may be merged to simplify the provenance graph. Another special situation is when the existing node c_i is referenced by a frame slide or a bookmark (see Section 4.3); in this case, the node has to be preserved and the rule cannot be applied as it would lead to broken references.

(3) If none of the rules above apply, a new node v with state q and an edge with type e have to be added to the graph, yielding $G_{i+1} = (V_{i+1}, E_{i+1}, v, s_{i+1}, t_{i+1})$ with

$$\begin{aligned} V_{i+1} &= V_i \cup \{v\} \\ E_{i+1} &= E_i \cup \{(c_i, v)\} \\ s_{i+1}(x) &= \begin{cases} q & \text{if } x = v \\ s_i(x) & \text{otherwise} \end{cases} \\ t_{i+1}(x, y) &= \begin{cases} e & \text{if } (x, y) = (c_i, v) \\ t_i(x, y) & \text{otherwise} \end{cases} \end{aligned}$$

These rules only make limited use of knowledge about the domain and other meta information, as the provenance system operates at a low level of abstraction. Accordingly, they may be applicable in other domains as well. In Section 3.2 more advanced and aggressive optimization techniques are discussed; in particular, motif-based aggregation seems promising for this use case (with rule (2) being a very simple variant of this technique). The issue with extending this technique to incorporate more advanced motifs (e.g. an alteration between add and modify events) is that selecting adequate motifs requires domain knowledge [MRS+13], and therefore probably user interaction (violating **R5**). Automated aggressive optimization risks culling relevant analysis steps from the tree (contradicting **R4**), and likely requires extensive user studies to find a good balance between optimization and usability.

4.2.3 Visualization

There must be a way to extract information from provenance for it to be useful. As discussed in Section 3.2, there are two major paradigms for querying provenance: directed and exploratory queries [CAB+14]. Directed querying is used if one knows what they are looking for and can formulate exact queries with a querying language. Exploratory querying, in contrast, is useful for gaining a broad overview about a provenance graph, usually by means of visualization.

The use case of storytelling in PRo3D favors the latter approach, as the user aims to gain an overview of a past analysis session to extract key points and build a coherent story. Usually, provenance graphs are visualized as node-link diagrams [BYB+13], but these kinds of graphs do not scale well with an increasing number of nodes [MS11]. Simplification strategies, as described in Section 4.2.2, remedy this issue by minimizing the graph. Techniques for simplifying the visualization itself (e.g. LoD approaches) further facilitate querying of the graph. Apart from the visual complexity of large node-link diagrams, the layouting algorithm has to be chosen to accommodate for limited screen space. Provenance in PRo3D has to be viewed and navigated, while displaying the 3D scene and the storyboard at the same time. Inline layouts require less space than traditional layouts, but are inadequate to represent trees with a large branching factor [KNF+01, HMSA08].

For storytelling in PRo3D, the design used in CLUE (see Section 2.5.6 and Figure 2.15) is an appropriate tradeoff to address these issues [GLG+16]. The layout algorithm makes use of the fact that the provenance graph is an arborescence to save space. The path from the root to the currently active node is kept in a single vertical line on the right side of the provenance view, giving easy access to previous analysis states. The provenance tree represents an (ongoing) analysis session, requiring users to orient themselves in relation to their current location. The vertical layout allows users to quickly locate their current position in the tree, and gives a quick overview about the steps that lead there. Branches are displayed to the left of this main trunk, leaving enough space for labels of the main nodes. The nodes themselves are rendered with multiple LoDs based on a DoI function, incorporating factors like the distance to the current node, if the node is part

of the trunk, or if it is bookmarked. The applied LoD is recomputed for each node on the fly, reducing the visual complexity of the provenance tree as the analysis progresses. For our implementation (see Chapter 5), we opt for a simple node-link diagram, since visualization algorithms for complex graphs is beyond the scope of this thesis.

4.3 Camera

For the DOM interpretation process in PRo3D, the position and orientation of the 3D camera play a special role. Strictly speaking, these properties are change artifacts according to the definition in Section 4.1. As the analysts investigate different locations and outcrops, they move the camera to view the relevant portion of the data. Likewise, the analysis of a single outcrop occurs at varying scales, requiring multiple camera locations and orientations. The state of the camera is tied to the various stages of the interpretation, which can be inferred from the camera to some degree. For example, if the camera is zoomed close to a surface, it can be inferred that the user is currently measuring and analyzing small-scale features of the corresponding outcrop. However, recording provenance of the camera is more problematic than for other properties of the visualization state. This section discusses issues related to camera provenance and possible solutions.

4.3.1 Camera as change artifact

The deliberations above suggest that the camera is supposed to be treated as a proper change artifact and tracked in the provenance graph. An issue with this approach is that the camera is adjusted frequently during an analysis, yet not all changes result in a new stage in the interpretation process. In an early prototype, we tested how camera movements can be recorded in a meaningful way. We found that the user adjusts the camera in-between actual analysis operations to either get an overview of the data, or find a better viewpoint for subsequent actions. As these adjustments are too small to be significant for the analysis itself, the provenance system should not be triggered by these events. Deciding whether a camera movement is relevant (e.g. the view is moved to another outcrop) or only qualifies as such an adjustment is difficult. Since provenance is supposed to be recorded automatically (**R5**) and available meta information in PRo3D is minimal, that decision has to be made based on the camera movements alone. For example, a distance metric can be applied to decide if the movement exceeds a certain threshold, determining if the change was significant enough to be stored in the provenance graph. Here, the obvious challenge is finding a metric and threshold that deliver results that are reasonable for the domain experts in all situations. If that threshold is too low, the provenance graph grows rapidly with camera events that are falsely deemed significant. The simplification rule (2) can remedy this if these movements happen consecutively, but our testing showed that small camera adjustments are often interleaved with other analysis actions. As already discussed, simplifying such cases requires a more advanced motif-based aggregation strategy, which is difficult to implement while satisfying

all requirements. In contrast, setting the threshold too high, or treating the camera as a view artifact altogether, results in missing provenance information. Navigating a provenance graph with insufficient data about corresponding camera positions and orientations becomes tedious; the difference between two states might not be apparent immediately, if it is limited to parts of the scene not visible with the current camera parameters (i.e. if there is a high variance in location or scale).

4.3.2 Characteristic views

Since finding a threshold that works in all situations is improbable or even impossible, approaches that require some user interaction have to be considered instead. Relying on user feedback for provenance risks distracting the user from their actual analysis tasks. Provenance solutions with such manual aspects have to keep the required interaction small, and build upon existing workflows. One approach is to extend the way PRo3D currently computes and stores *characteristic views* for every annotation. Characteristic views are a concept used in 3D object recognition [ADV06], and are basically 2D views of an object aiming to represent it as accurately as possible. In PRo3D this concept is slightly adapted, storing camera positions and orientations as views from which the corresponding annotation is visible. The user can quickly navigate to any annotation by clicking on an icon next to its name in a list, which moves the camera to the stored view. The characteristic view is computed based on the assumption that the camera is positioned favorably at the moment the user digitizes the annotation. Thus, when an annotation is added to the DOM, the current camera location and orientation are simply stored as its characteristic view. A similar idea can be applied to automatically store camera parameters in the provenance graph. Instead of keeping track of changes to the camera itself, the current camera properties are stored whenever an event triggers the provenance system. For example, if the user moves the camera, provenance is not updated immediately; if they subsequently modify the color of an annotation, resulting in a new or updated node in the provenance tree, the updated camera location and orientation are stored in that node as well. That is, camera parameters are not treated as change artifacts that trigger events on their own, but implicitly updated whenever the provenance graph is updated. Like for characteristic views of annotations, the premise is that the camera parameters at the time of an event are appropriate to view its results. Unfortunately, this premise is not true in all situations, as can be observed when large-scale polyline annotations are added to the DOM. Suppose, a geoscientist needs to annotate a bedding plane that extends multiple meters across the outcrop. To place the individual points of the annotation, they have to zoom close to the surface of the outcrop to discern fine details. After they place a point, they pan the view a few centimeters along the bedding plane to place the next one, repeating this process until the polyline is complete. At that point, the finished annotation is digitized, firing the corresponding event and triggering the provenance system. However, the camera is still zoomed in on the very last point of the polyline, which is not a suitable characteristic view of the whole annotation. The issue about minor adjustments made to the camera during analysis is also relevant for this implicit approach. Every small change to the camera is persisted in the provenance graph

as soon as an event is processed. While this does not increase the size and complexity of the graph, it can make navigating to past states irritating. A small difference in camera position is likely to be irrelevant for the analysis, and distracts the user from identifying the actual difference between two states. A possible solution to this problem is letting the user decide if they want to restore the camera parameters when navigating to a past state. For example, to restore a saved view, the user has to hold an additional key when clicking on the node in the provenance graph. This approach requires no user interaction during the analysis itself, but also offers no means for the user to amend unreasonable views that are stored automatically.

4.3.3 Bookmarks

Another approach to handle camera views is to build upon the bookmark feature that is available in PRo3D. Bookmarks allow the user to save, manage, and revisit interesting and informative views of the scene. Users further employ bookmarks as a crude substitute for sophisticated storytelling features, allowing them to quickly navigate a scene and present key data points. Bookmarks can be extended to not only contain camera properties, but also reference an analysis state in the provenance graph (similar to a slide in a story). Visiting such a bookmark restores the corresponding state in addition to moving the camera to the stored view. With this system in place, provenance can treat the camera as a view artifact, avoiding the issues discussed in the previous sections. In turn, the user is responsible for managing the camera, e.g. by creating a bookmark whenever a milestone is reached. This requires the greatest amount of user intervention with provenance of all the approaches discussed in this section, but still only builds upon the existing workflow of managing bookmarks in PRo3D (conforming to **R5**). Like with characteristic views, users can decide themselves if they want to restore the associated view, when visiting a past analysis state. Either they navigate to the desired state by interacting with the provenance graph directly, or select the corresponding bookmark, which also contains the stored view. The biggest advantage over implicitly handling the camera is that the user can decide what camera parameters constitute a characteristic view for a given state, and may also modify it at a later time.

Our solution builds upon this bookmarks-based approach, since it is the most flexible one and works well in most situations. In the next chapter, we describe how bookmarks, provenance, and the storyboard can be combined in the context of visualization design.

CHAPTER 5

3D geological stories

Based on the deliberations of the previous chapter, we implemented a provenance-based storytelling prototype and integrated it into PRo3D. In this chapter we present this implementation, discuss practical design choices (especially from a software engineering perspective), and demonstrate the capabilities of our solution by reproducing a traditional geoscientific story entirely in PRo3D. Source code and formal definitions are presented in F# rather than a mathematical notation, since it is more concise and PRo3D (and thus the prototype) is implemented in this language.

5.1 Visualization design

In this section, we present the user interface of our prototype and discusses design decisions thereof. Figure 5.1 shows the GUI with a short description of its elements. The top menu bar (1) gives access to basic features of PRo3D such as the annotation drawing tool and a menu for changing the camera mode. The render view (2) shows the 3D scene allowing the user to navigate the data and view the outcrops from any angle. Our prototype augments the render view with a 2D overlay to create, view, and edit frame slides of the story. The history of the current analysis session is visualized as a tree in the provenance view (3). The story itself can be inspected and modified in the storyboard view (4) by adding, removing, and editing slides. The sidebar to the right of the render view (5) provides means to modify and group annotations and surfaces. Moreover, bookmarks of the current view and analysis state can be created and given a descriptive name to easily access milestones of the analysis at a later stage. In the following sections, we discuss design details of the GUI elements added through our prototype.

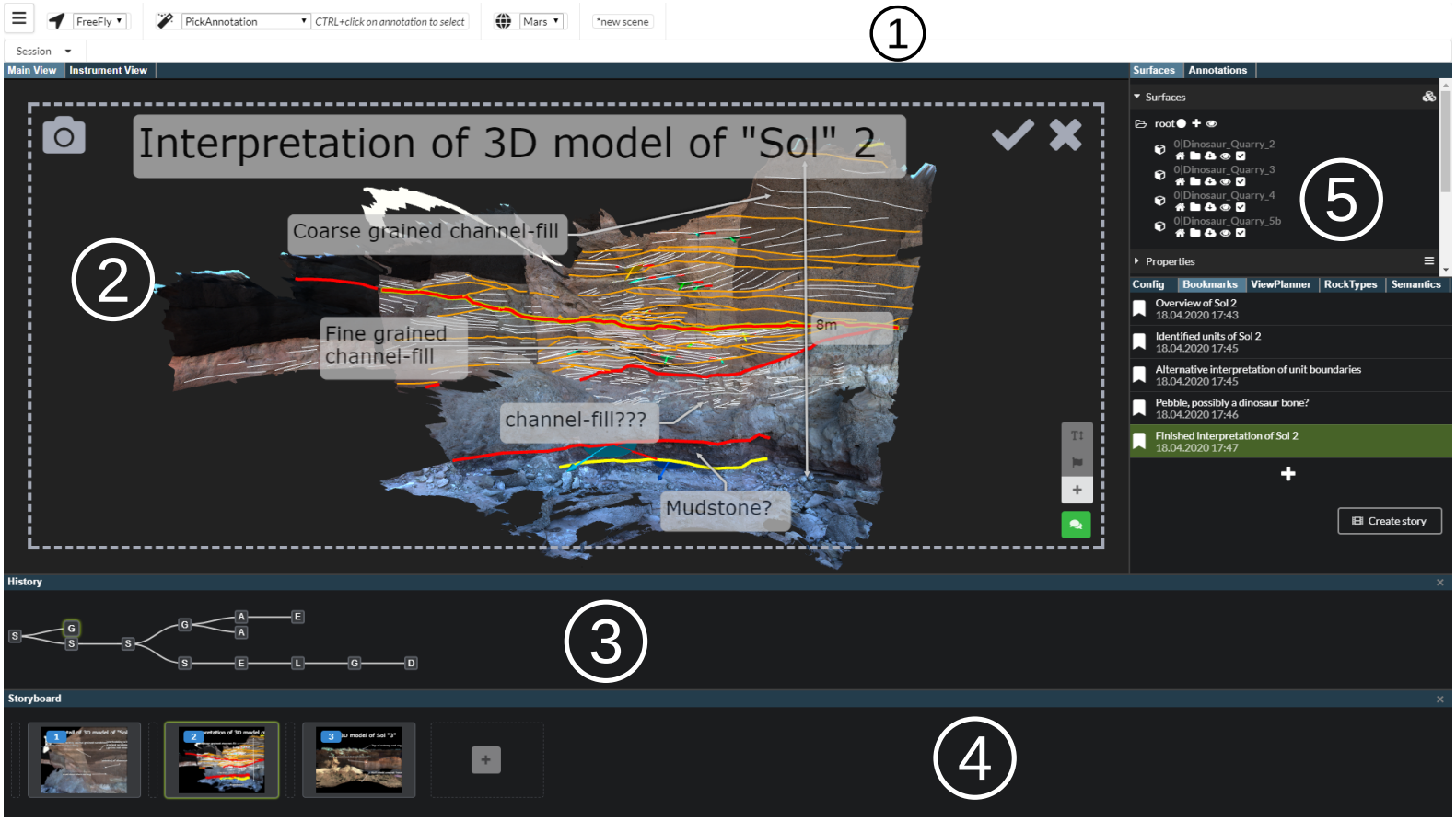


Figure 5.1: Overview of the user interface of PRo3D with storytelling features. It consists of (1) a toolbar showing basic PRo3D tools and options, (2) the render view displaying the 3D scene with a 2D overlay for storytelling, (3) a simplistic representation of the provenance tree, (4) the storyboard showing the slides of the current story, and (5) a sidebar providing more advanced PRo3D features such as organizing surfaces, annotations, and bookmarks.

5.1.1 Overlay

The main render view of PRo3D shows the 3D scene from the current camera position. Our prototype reuses this window to allow users to view and edit frame slides of the story. Whenever a frame slide in the storyboard is selected, the camera moves to the associated position and orientation, and the referenced analysis state is restored. The application is now in authoring mode, enabling users to make changes directly to the current slide; this is indicated by the gray dashed border around the render view and the camera icon in the top left corner. The checkmark and cross icons in the top right corner are confirm and cancel buttons to leave the authoring mode, deselecting the slide and returning to the regular analysis mode. The former is only visible if changes were made to the camera or analysis state; clicking it results in the made changes being saved to the selected slide. In contrast, the cancel button is always visible, and discards any changes made to the slide upon activation. This interaction constitutes an implicit transition between the exploration and authoring stages, whereas CLUE requires the user to explicitly switch to the desired stage by pressing a button. As a result, there are effectively only two storytelling stages in our prototype: a combined exploration and authoring stage, and a separate presentation stage. As touched upon in Section 2.2, there may be multiple occasions at which one has to return to the exploration stage from the authoring stage to gather additional facts for the story. Our approach facilitates this by minimizing the number of user interactions required to do so.

Apart from camera and analysis state information, the overlay also displays annotations of the selected slide. Story annotations consist of a text label and any number of anchor points in the 3D scene. As discussed in Chapter 4, annotations without any anchors describe general properties of the scene, whereas anchored annotations explain specific features. Point annotations are anchored annotations that describe one or multiple (but separate) points in the scene. Single arrows connect the label and each of the anchor points. Line annotations are another form of anchored annotation with exactly two anchor points. A double arrow connects the two anchors, the optional label is positioned in the middle of them. Figure 5.2 shows the different types of annotations available in our prototype. The user can position general and point annotations explicitly within the 2D slide, giving them control over the final layout of the slide. We decided to let the user determine the layout of these elements, as they contain the main information of the story. In contrast, line annotations are mainly used to denote the extent of certain geological features and are only secondary to the story. Consequently, annotations of this type are not explicitly positioned within the slide, but defined by the 3D positions of their anchors. Thus, the user controls the layout of the main annotations, while elements conveying additional information are positioned automatically. A disadvantage of a fixed 2D layout is that navigating the scene can be cumbersome if the 2D annotations cover the render view. Therefore, the visibility of the slide annotations can be toggled by pressing the green button in the bottom right corner of the overlay. This button also reveals a menu that lets the user add more annotations and edit properties of existing ones (e.g. the font size).

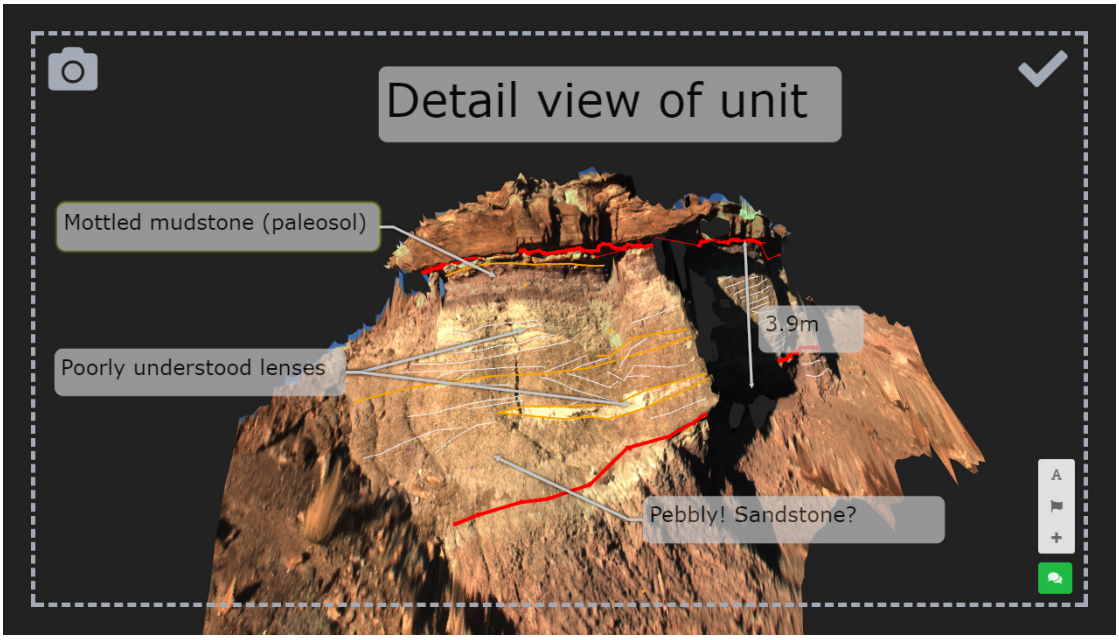


Figure 5.2: Example slide with general annotations and multiple variants of anchored annotations. General and point annotations are positioned explicitly within the 2D slide, line annotations are positioned automatically according to their two anchors.

5.1.2 Brushing and linking

Both the storyboard and the bookmarks window show data that reference provenance information presented in its own window. Interactions and data in those windows are associated via brushing and linking.

The provenance window in our prototype uses a node-link diagram to visualize the provenance tree. The tree layout is computed using D3.js [BOH11], which implements the Reingold-Tilford 'tidy' algorithm [RT81]. Our prototype does not employ any LoD-based approaches to handle large graphs, since scalable tree layout algorithms are not the focus of this thesis. Instead, we refer to CLUE by Gratzl et al. [GLG+16], which includes a scalable provenance tree as discussed in Sections 2.5.6 and 4.2. In our prototype, each node is visualized as a simple glyph and labeled with a character representing the event that lead to the corresponding state. The currently selected node is highlighted with a green outline, hovered nodes have a blue outline. The storyboard view uses the same color scheme to indicate if a slide is currently hovered or selected. Users can build their stories with the storyboard by adding, removing, moving, and duplicating slides. For each slide a thumbnail is stored and shown to summarize its content. Clicking a slide selects it and restores its state and camera parameters, allowing the user to modify it as described above. Figure 5.3 shows the bookmarks view in more detail, which lists bookmarks set by the user in chronological order. Bookmarks allow the user to save and revisit milestones of the analysis by giving them a descriptive name. The associated view of a bookmark can

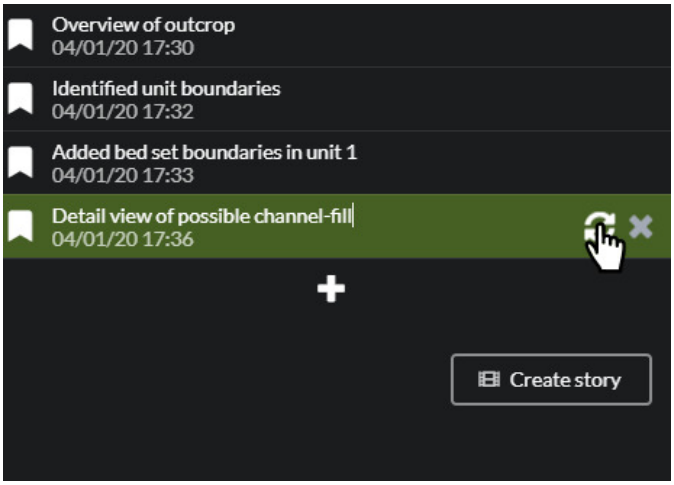


Figure 5.3: The bookmarks view lets the user save and manage milestones during the analysis. Bookmarks can be given a descriptive name, modified, and allow the user to revisit the associated state. Bookmarks can also act a basis for the finished story.

be adjusted at a later point by pressing the update button. Our prototype also includes a feature introduced in CLUE that lets users automatically generate a preliminary story based on the bookmarks. The rationale is that users create bookmarks of important analysis stages during the exploration phase; a complete story of that analysis is likely to include these milestones, making the bookmarks a convenient starting point for the story. To this end, the user may press the "Create story" button, which automatically creates a slide for each bookmark with its description as an annotation.

Hovering over a slide or a bookmark in their respective views also affects the provenance view. The node referenced by the hovered element is highlighted with a blue outline as if it was hovered itself (see Figure 5.4). This interaction also works vice-versa, i.e. highlighting a slide or bookmark whenever the referenced node is hovered in the provenance view. It is also possible to discern which nodes in the graph are currently being referenced. This is indicated by a lighter fill color as seen in Figure 5.4. Both story and bookmark references are highlighted in the same way to reduce the visual complexity of the provenance view. However, references are only shown if the corresponding view is being hovered. For example, the user has to hover over the storyboard to see which nodes are being referenced in the story. This allows the user to quickly make out which nodes are referenced by elements in both views respectively. The decision of which type of reference to show can also be tied to which views are currently visible in the application. For instance, if only the storyboard and provenance views are visible, it does not make sense to show bookmark references. Our prototype does not implement this logic, since PRo3D currently does not offer a way to easily close and reopen views.

Figure 5.4: Interaction between the provenance and storyboard views via brushing and linking. Hovering a slide will also highlight the referenced node in the provenance graph. Nodes that are referenced by a slide are indicated with a light fill color.

5.2 Results

We presented our prototype to our primary collaborator, a sedimentologist from the Imperial College of London (ICL) involved in the scientific dissemination of outcrop data from the NASA Curiosity Rover. He had the opportunity to experiment with a fully functional version of our prototype at two workshops. Both times, he was able to construct a short story from annotation states in the provenance graph and replay it in the presentation mode. In general, his feedback was very positive and "he can't wait to use this in PRo3D". Furthermore, we presented an early version of our prototype at a planetary science workshop hosted at ICL, in the form of a video to show the ongoing research in PRo3D. The prototype itself and the concept of illustrating scientific analyses directly in the application it was performed, were well received by planetary scientists.

In the remainder of this section we present the capabilities of our prototype by reproducing an actual geological story. We intend to demonstrate the accuracy (and limitations) of our prototype when it comes to producing proper geological stories using real data. The original slide show by Marijn van Cappelle is based on data from the Hanksville-Burpee Dinosaur Quarry (HBDQ) incorporating manually annotated screenshots of the dataset visualized in PRo3D. It consists of three parts:

1. Introductory slides briefly describing the location, how the data were obtained, and the rationale of the HBDQ project.
2. Data slides showing the interpretation of the various outcrops. Each slide presents an outcrop imaged during a "sol", a (simulated) Martian day, augmented with annotations that geoscientists added either directly in PRo3D, or by manually editing the screenshot using image editing tools.
3. Concluding slides presenting a preliminary correlation and mapping between the outcrops based on the individual interpretations. These figures were created outside of PRo3D and added as images to the slide show.

The focus of this thesis is how original 3D data can be incorporated into geological stories. As such, our prototype does not support formatted text slides with imported images, which are required for the introductory slides. Moreover, PRo3D does not currently support the creation of correlation logs as contained in the concluding slides. Consequently, we omitted these slides and only recreated the main content of the slide show, i.e. the data slides showing the interpretations of the individual outcrops. Figures 5.5 to 5.12 show our results compared to the corresponding original slides. The screenshots of our results show how the geological story is visualized in our prototype's presentation mode. In contrast to the authoring mode, the text labels are drawn as simple outlined text without any background color to minimize occlusion of the geoscientific data. Our slides are almost identical to their original counterparts, demonstrating the potential of our approach to produce real geological stories. In addition to the aforementioned limitions of our prototype, Figures 5.7 and 5.10 to 5.12 show that our results are missing sedimentary

logs (see Chapter 4), which are used in the original presentation. These diagrams were created separately, as PRo3D does not support the creation of such diagrams at the time of writing. However, future versions of PRo3D will support the generation of sedimentary and correlation logs, bridging the gap between our storytelling solution and traditional geological presentations.

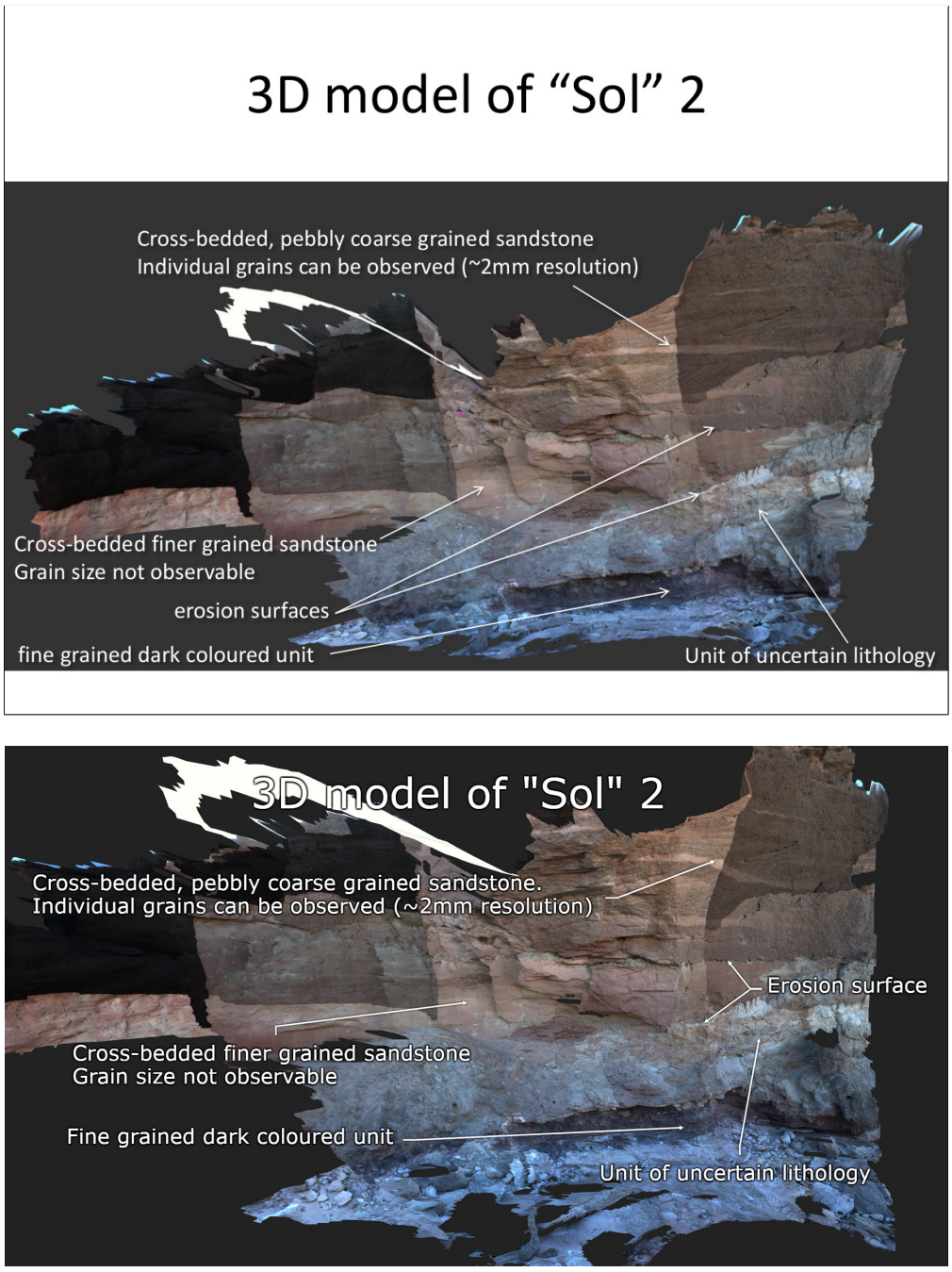


Figure 5.5: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

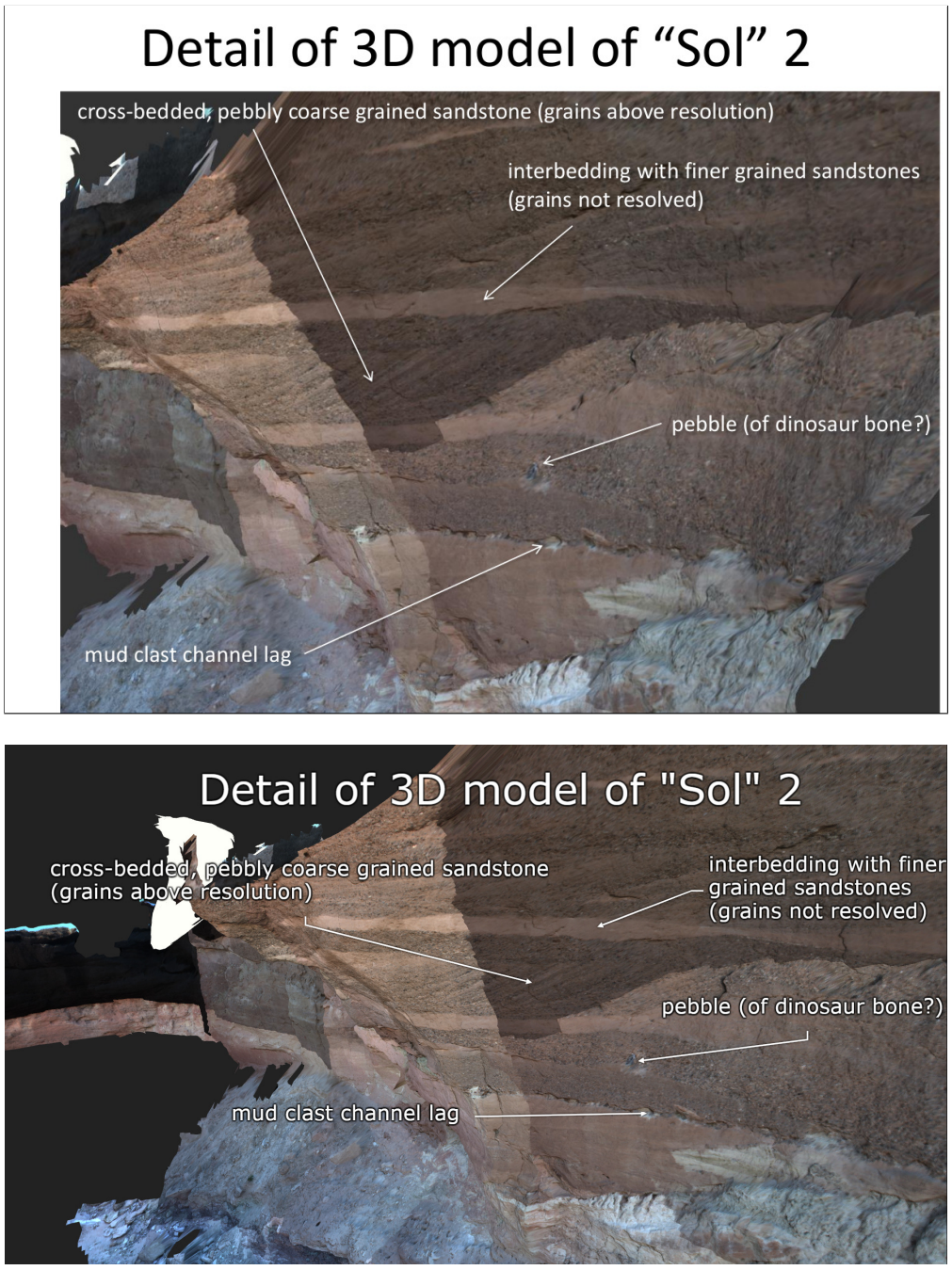
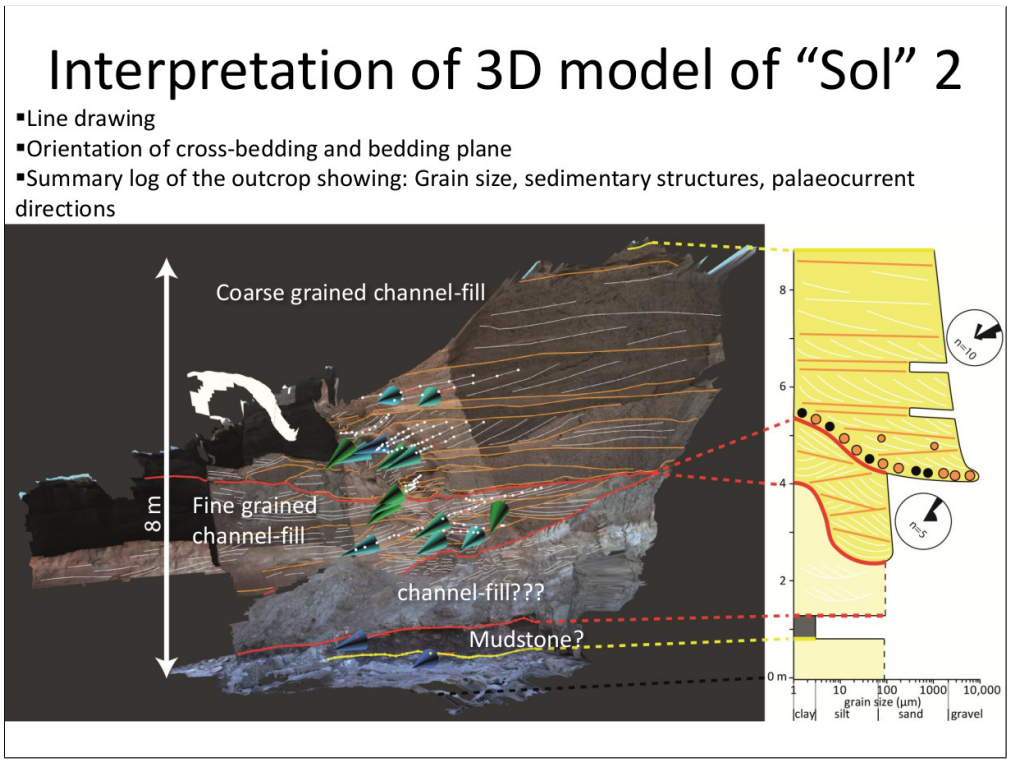


Figure 5.6: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016



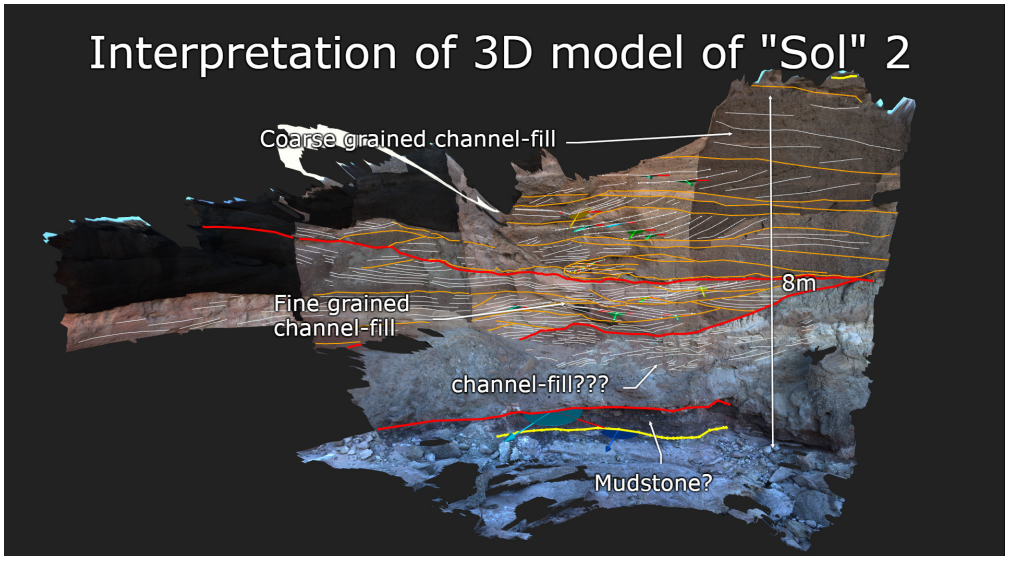


Figure 5.7: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

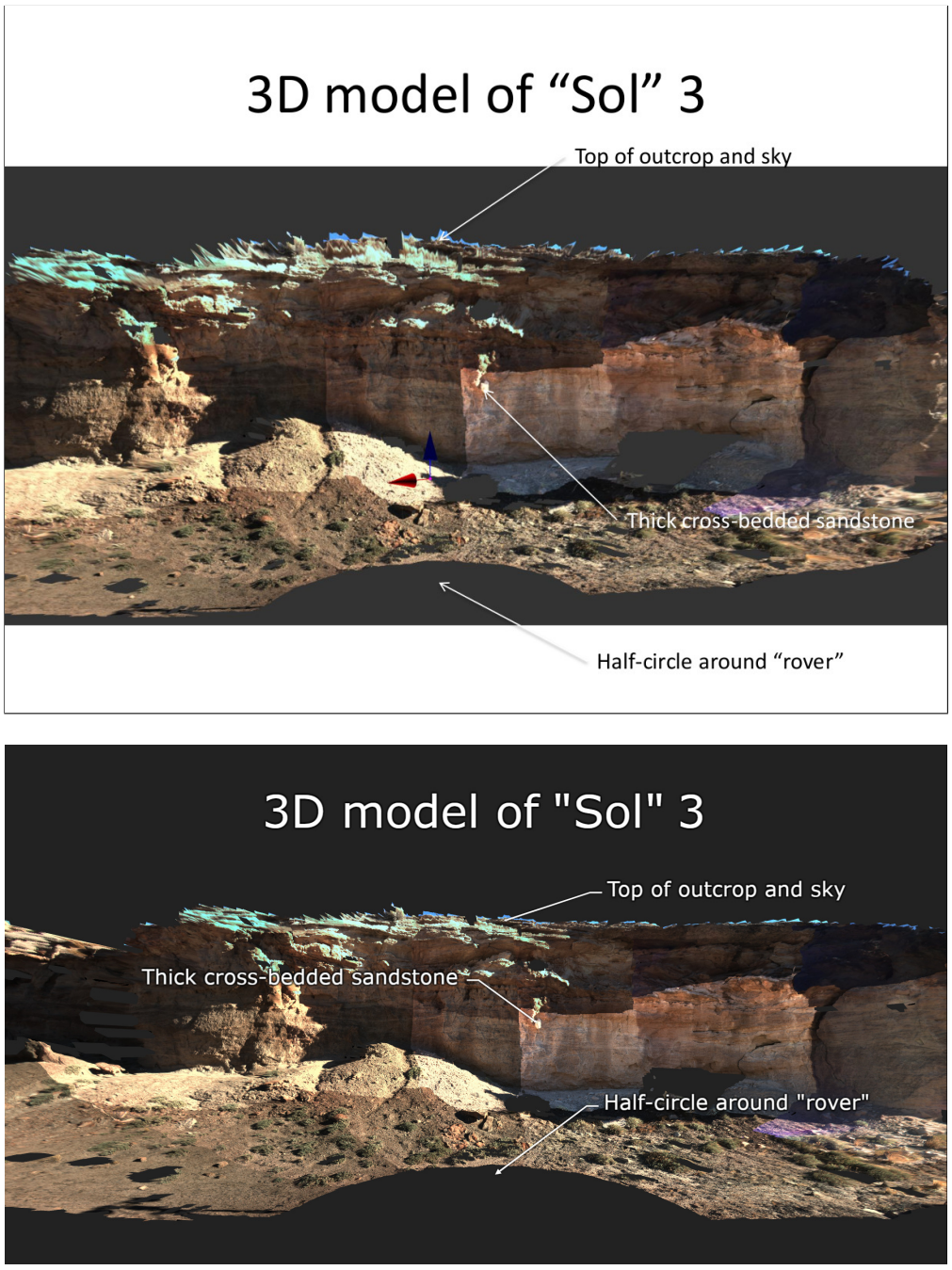


Figure 5.8: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

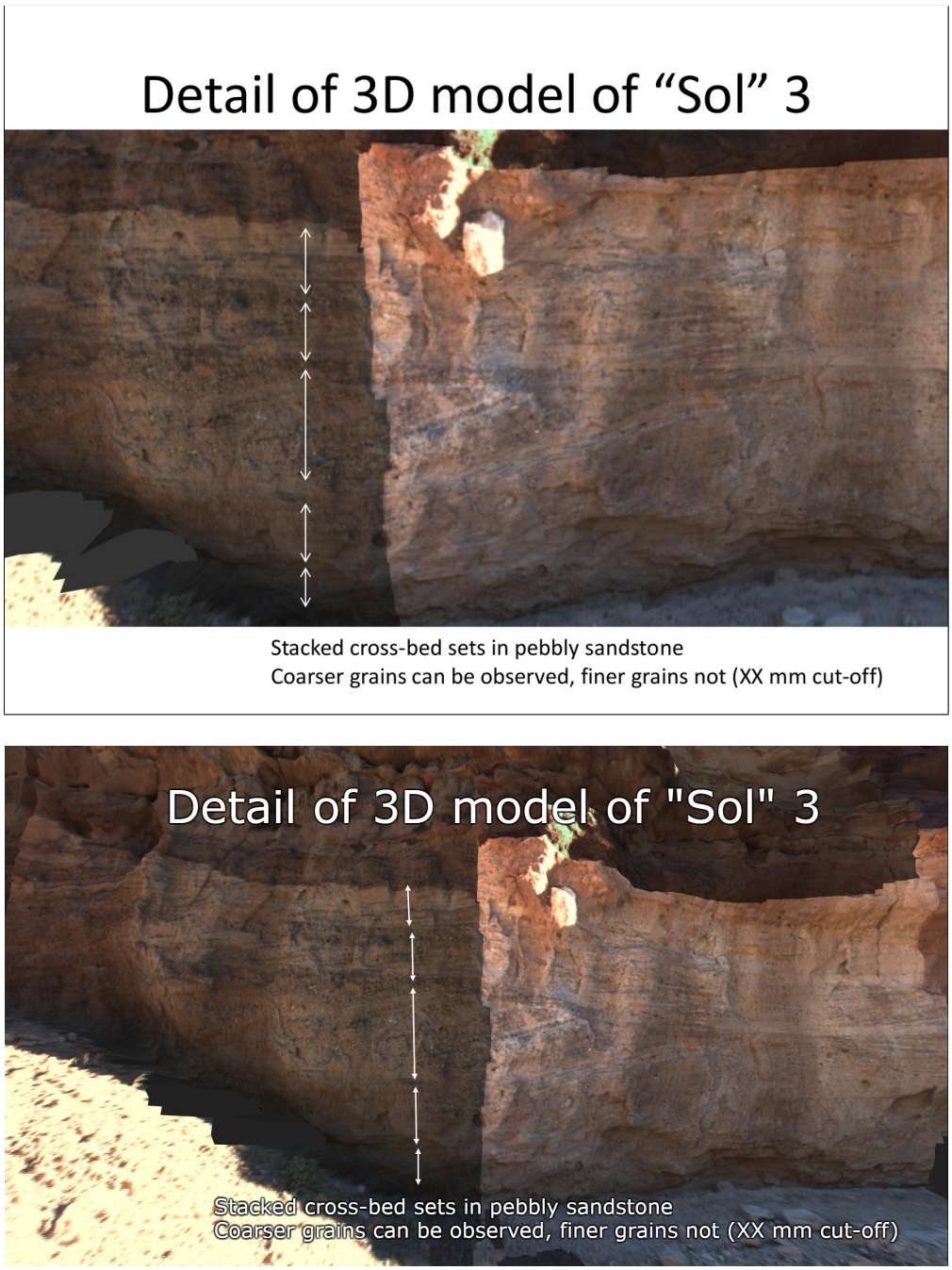
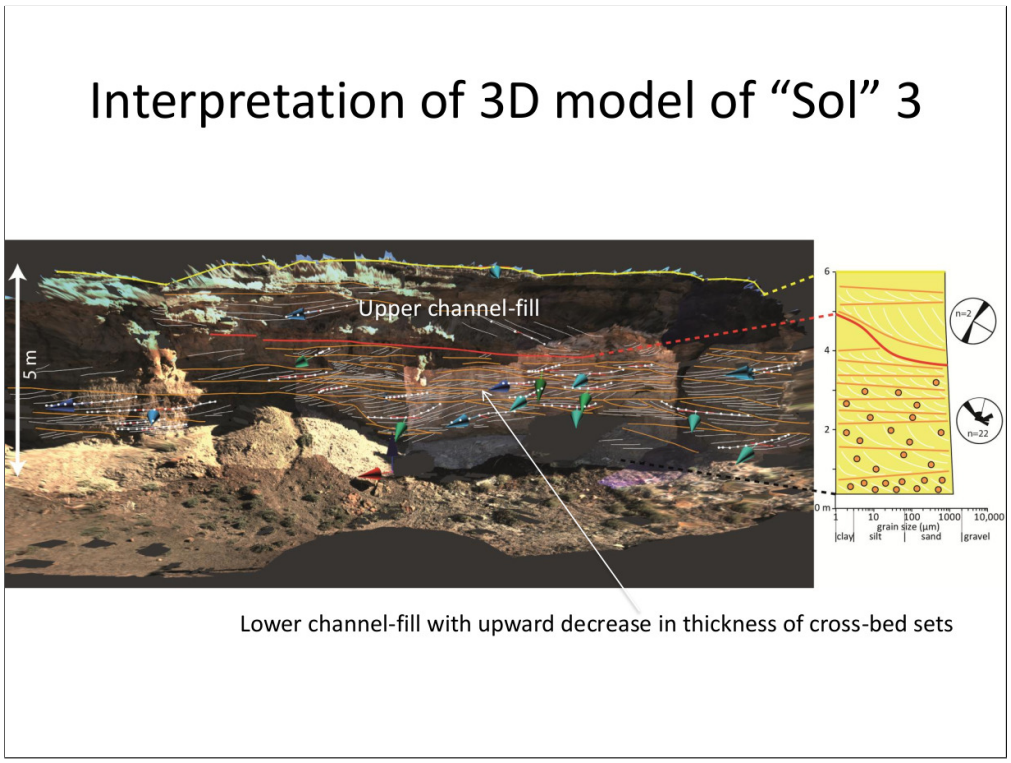


Figure 5.9: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016



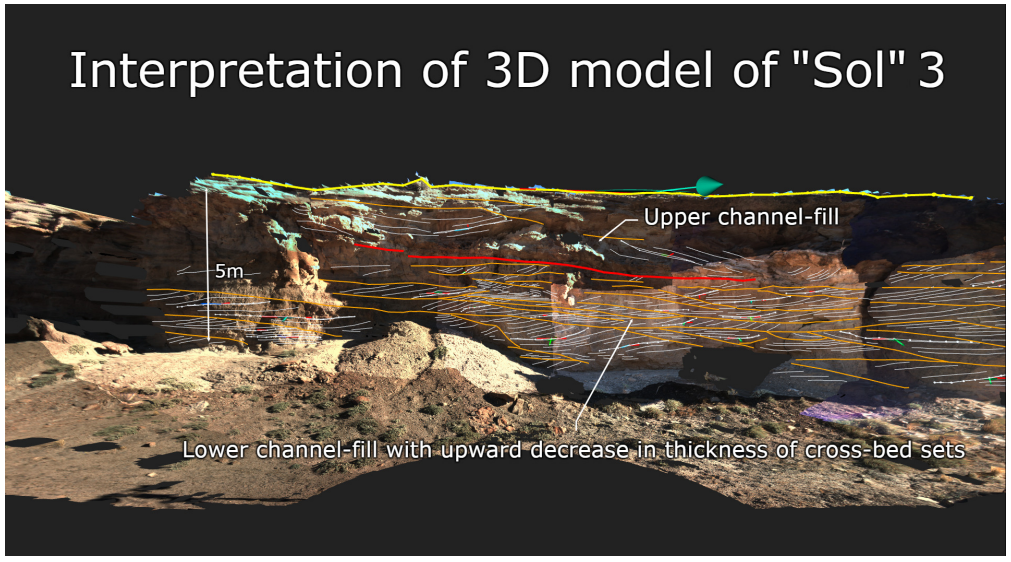


Figure 5.10: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

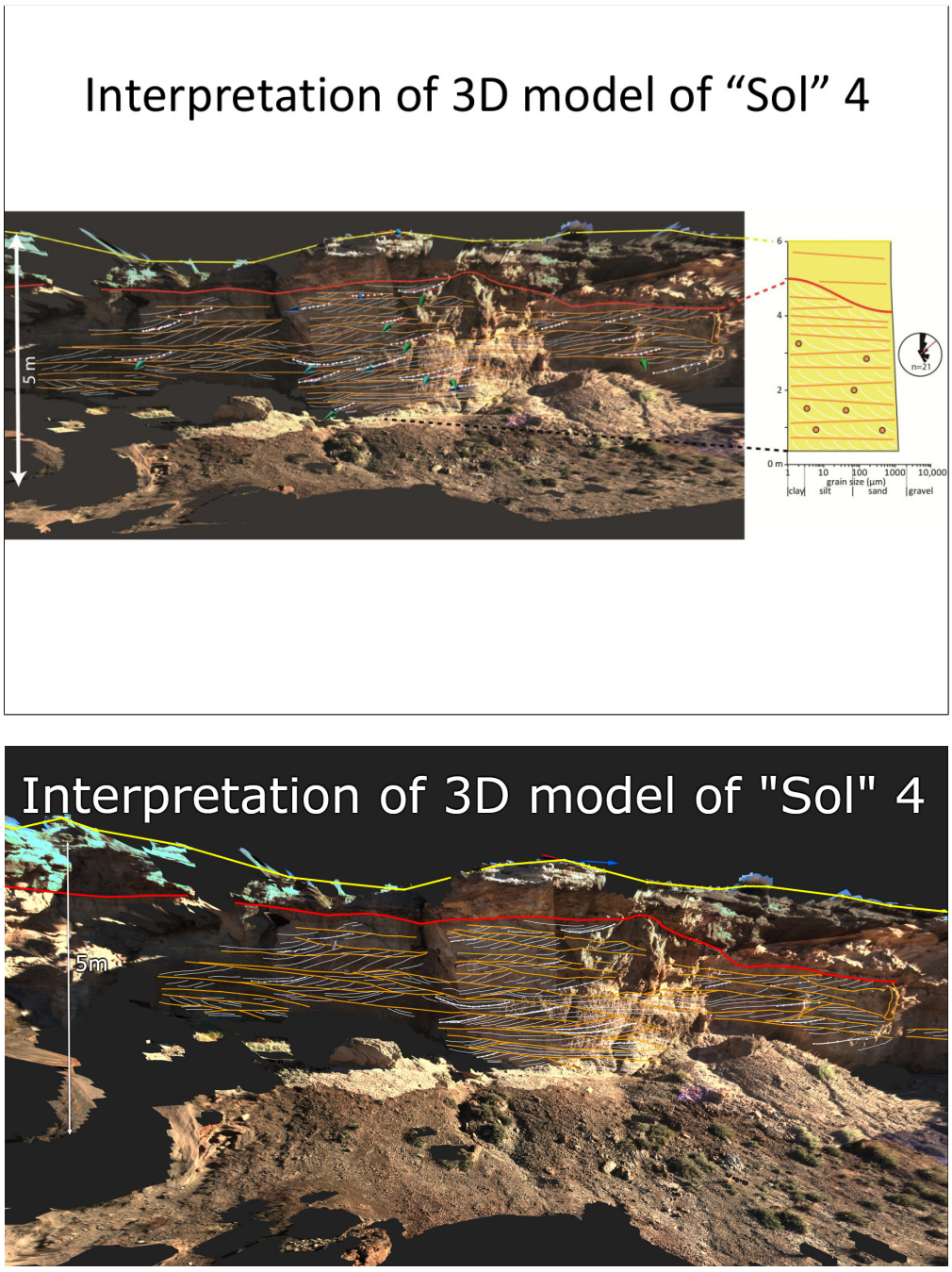


Figure 5.11: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

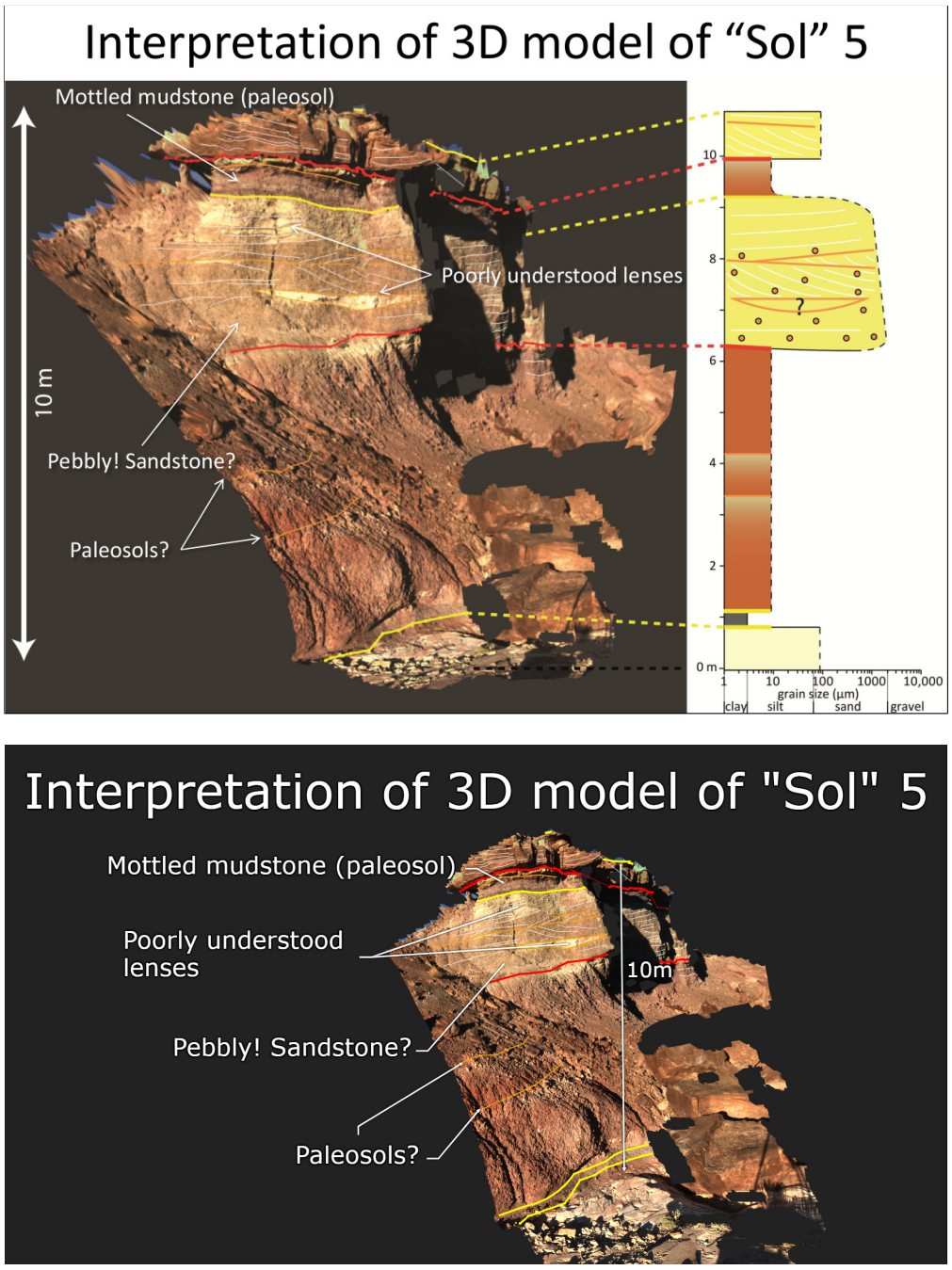


Figure 5.12: Slides showcasing data from the HBDQ project (top), recreated in our storytelling prototype (bottom).

Source: Original slides by Marijn van Cappelle 2016

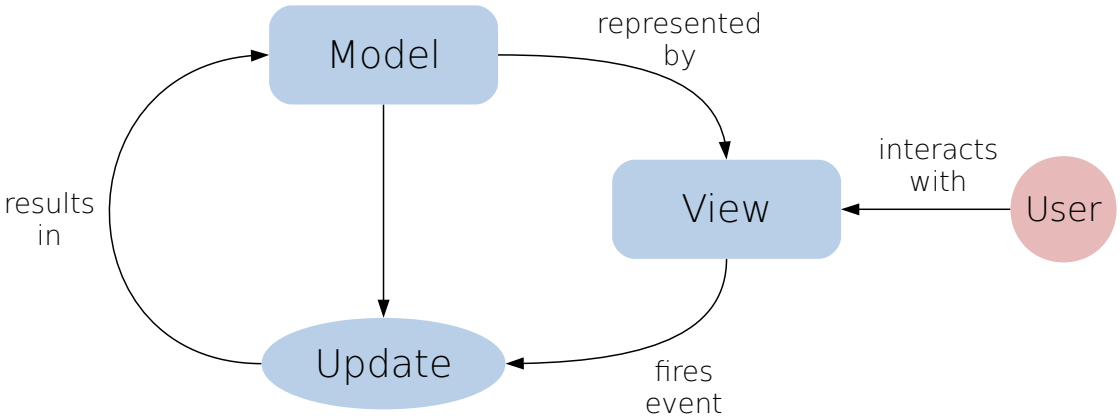


Figure 5.13: The Elm Architecture consists of the model, the view, and the update logic. The model is the current state of the application, the view its visual representation, and the update logic processes events from the view to compute a new model.

5.3 Implementation

PRo3D is based on Aardvark (An Advanced Rapid Development Visualization and Rendering Kernel) [Aarb], a collection of libraries for visual computing and real-time rendering. It includes an incremental rendering engine to render dynamic, complex scenes in an efficient manner [HSMT15]. The cost of updating a scene within this incremental system is independent of its overall complexity, but only depends on the scope of the changes. This concept is extended to parts of Aardvark unrelated to rendering, enabling developers to build applications that take advantage of incremental evaluation.

Media [Aara] is an Aardvark library that builds upon this incremental engine allowing developers to build modern GUIs for their applications in a purely functional and declarative manner. It is inspired by Elm, a language designed and used to create web-based GUIs [Cza12]. At its core is the Elm Architecture [Elm], a functional design pattern that consists of three aspects: the model, view, and update logic. The model holds the current state of the application, whereas the view is the visual representation of the model (i.e. the user interface or the rendered 3D scene). The user interacts with the view (e.g. presses a button or selects an object in the scene), which triggers events that are handled by the update logic. The update logic is essentially a function `val update : Event -> Model -> Model` that maps the current model and an event to an updated model, where `Model` and `Event` are the formal types of models and events respectively. Figure 5.13 illustrates this pattern. In Media this architecture can be extended in a hierarchical fashion. The model consists of multiple sub-models, which in turn are made up of smaller models. All these models provide their own view and update logic, resulting in an application divided into a hierarchy of independent modules. The top-level update function delegates its work to the other update functions based on the processed event. Listing 5.1 shows a minimal example for such a hierarchy.

```
type Model = {
    a : A.Model
    b : B.Model
}

type Event =
    | EventA of A.Event
    | EventB of B.Event

let update (e : Event) (m : Model) =
    match e with
    | EventA x ->
        { m with a = m.a |> A.update x }
    | EventB x ->
        { m with b = m.b |> B.update x }
```

Listing 5.1: Example of a hierarchical model and update function.

The view representation is independent of the rest of the application logic; Media employs a representation based on web technologies (HTML, CSS, JavaScript, etc.) at the time of writing. The Document Object Model tree generated from the model can be viewed with Aardium (a wrapper for Electron [Ele]) as a desktop application or in a web browser. This way Media facilitates the development of cross-platform applications, which can use specialized JavaScript libraries such as D3.js [BOH11]. The incremental engine ensures that the DOM tree is updated in an efficient manner; as the model gets modified only the parts of the view that are affected by the changes have to be recomputed.

We exploit this architecture to integrate our prototype into PRo3D in an unintrusive way. As discussed in Chapter 4, PRo3D is still being actively developed, with features being changed or added in the future. Since the storytelling features are reliant on the main application, a tight coupling would incur a considerable maintenance overhead as PRo3D's development continues. Instead, we integrated the storytelling and provenance mechanisms as separate modules that use a well-defined interface to communicate with the main application. This minimizes the additional work required to keep the storytelling features up-to-date with the rest of the application. As a result, the continued development of the analysis toolset in PRo3D is not impeded by our prototype. The hierarchical Elm Architecture employed by Aardvark and PRo3D facilitate such an integration. Figure 5.14 shows the architecture of our prototype and how it interacts with PRo3D. It consists of a top-level application that follows the hierarchical Elm-style model-update-view pattern. Its main components are the PRo3D, provenance, and storytelling modules, which are all Elm-style applications themselves. The top-level application is necessary to combine all the sub-applications by delegating update calls (as seen in Listing 5.1), and combining

```
type Model = {
    pro3d       : PRo3D.Model
    story       : Story.Model
    provenance  : Provenance.Model
}

type Event =
    | PRo3DAction       of PRo3D.Event
    | StoryAction       of Story.Event
    | ProvenanceAction  of Provenance.Event
```

Listing 5.2: Model of the top-level application of our prototype.

views into a complete representation. Figure 5.14 and Listing 5.2 show a simplified model with the three main modules, yet the prototype also includes a range of other modules (e.g. for handling animations and user sessions). PRo3D itself consists of a multitude of modules, and does not have any notion about the top-level application. It is isolated from the prototype modules, ensuring that only minimal changes to PRo3D itself are required for the integration. The disadvantage of this design is that PRo3D cannot draw from the code base added by the prototype. For example, code that handles camera animations for story playback cannot be invoked from PRo3D and needs to be implemented separately.

Both the storytelling and the provenance modules require access to properties contained within the PRo3D model (e.g. to read and modify the camera position). The most straightforward way would be to let the top-level `update` function pass the `pro3d : PRo3D.Model` field directly to the `Story.update` and `Provenance.update` functions. However, this would lead to a tight coupling between these modules and the PRo3D module; every change to the `PRo3D.Model` type would necessitate adjustments at multiple locations within the prototype, resulting in a considerable development overhead. Instead, our prototype provides an abstraction layer that handles access to the PRo3D model. For example, it provides the functions

```
val getCamera : PRo3D.Model -> CameraView
val setCamera : CameraView -> PRo3D.Model -> PRo3D.Model
```

to retrieve and set the current camera in PRo3D respectively. As the `PRo3D.Model` type is modified, the implementation of these functions has to be adjusted. Yet, as long as the interface of the abstraction layer remains consistent, updates to other parts of the storytelling prototype are not required. The abstraction layer is also responsible for translating the PRo3D model to the internal representation used by the provenance module. As discussed in Section 4.2, the nodes of the provenance graph represent different states of the change artifacts. The change artifacts, however, are nested deeply in the hierarchy of the PRo3D model and need to be converted to a concise representation before being stored. Otherwise, provenance of other properties would be implicitly stored

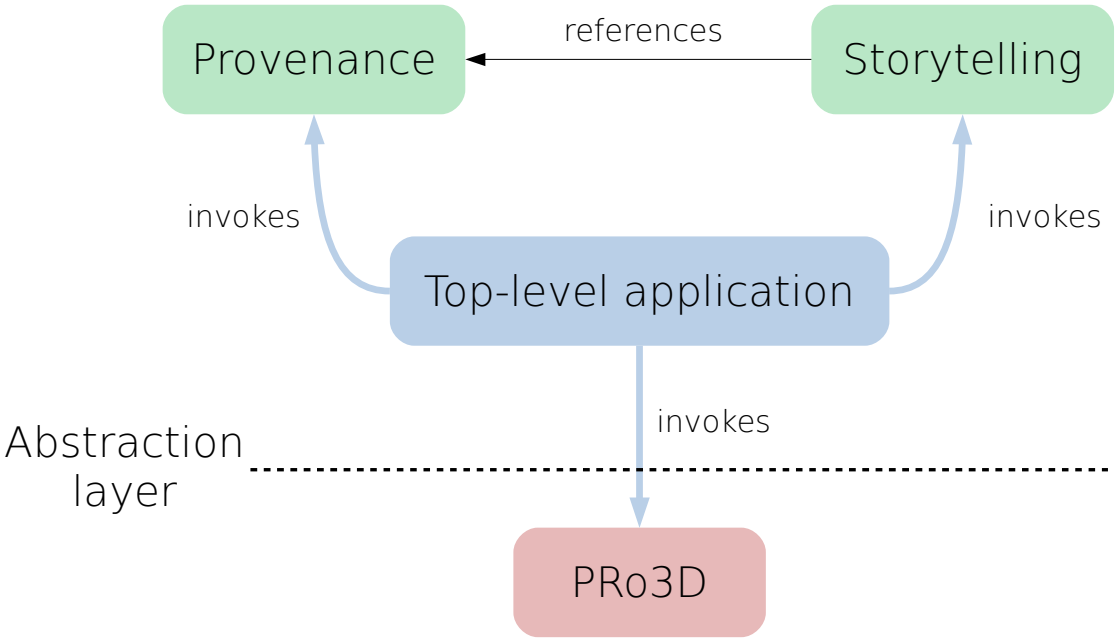


Figure 5.14: Simplified depiction of our prototype's system architecture. It consists of a top-level application that combines its three main modules: provenance, storytelling, and PRo3D. The abstraction layer provides a unified interface for the top-level modules to interact with the PRo3D application.

and restored, while also increasing the memory requirements of the provenance graph. To this end, the abstraction layer provides a pair of functions

```
val reduce : PRo3D.Model -> Model'
val restore : PRo3D.Model -> Model' -> PRo3D.Model
```

where `Model'` contains the state of change artifacts:

```
type Surfaces' = { ... }

type Annotations' = { ... }

type Model' = {
    surfaces : Surfaces'
    annotations : Annotations'
}
```

The `reduce` function takes a parameter of type `PRo3D.Model` and returns the corresponding simplified version of type `Model'`. The `restore` function handles the reverse conversion, applying the values of the change artifacts contained in the `Model'` parameter to the given full model. Apart from the application state, the provenance graph also

stores the events that lead from one state to another one. Like states, events are not saved directly, but converted to an internal representation:

```
type Event' =
    | AddAnnotation
    | EditAnnotation
    | DeleteAnnotation
    | LoadAnnotations
    | GroupAction
    | EditSurface
    | Unknown
```

Application events are converted to this representation with a function

```
val reduceEvent : State' -> State' -> PRo3D.Event -> Event'
```

that takes two simplified states and the original event that lead from the first state to the other one, to compute the corresponding simplified variant. If there is no such correspondence (e.g. both states are equal), `Unknown` is returned and the event is ignored for provenance. The abstraction layer and the purely functional, hierarchical architecture enable provenance to be processed without directly accessing the PRo3D model. Listing 5.3 shows the corresponding portion of the top-level `update` function. Whenever a PRo3D event `a` is processed, the model of the main application `s` is updated by invoking `PRo3D.update`, yielding its next state `t`. These two models are reduced with `reduce` to gain their simplified representations `s'` and `t'`. Likewise, the current event `a` is simplified by calling `reduceEvent` resulting in `a'`. The three reduced values `s'`, `t'` and `a'` are passed to `Provenance.update`. The provenance graph is processed and updated employing the strategies discussed in Section 4.2.2, resulting in the updated provenance model `p`. Finally, the `pro3d` and `provenance` fields of the top-level model are replaced with the updated values.

The storytelling module follows this separation of PRo3D model and internal representation as well. Stories merely reference the nodes of the provenance graph, and the main application is controlled via the abstraction layer (e.g. adjusting the camera in the PRo3D model). The view representation of the stories is separate from PRo3D's view as well. Even though story annotations are anchored in the 3D scene, the storytelling module does not alter the scene graph of the main application. Instead, the annotations are displayed using an independent 2D overlay that is placed on top of the main render view.

```
let update (e : Event) (m : Model) =
    match e with
    | PRo3DAction a ->
        // Update PRo3D
        let s = m.pro3d
        let t = s |> PRo3D.update a

        // Reduce models and event
        let (s', t') = (reduce s, reduce t)
        let a' = reduceEvent s' t' a

        // Process provenance with reduced values
        let p = m.provenance |> Provenance.update s' t' a'

        { m with pro3d = t
                 provenance = p}

    | StoryAction a ->
            ⋮
```

Listing 5.3: Top-level update logic for processing PRo3D events and provenance. The abstraction layer allows provenance to be managed without directly accessing PRo3D data structures.

CHAPTER 6

Conclusion

In this thesis, we explored how storytelling can be applied to geological analyses that are conducted in PRo3D. The goal is to facilitate knowledge transfer, which traditionally relies on slide shows created in separate programs such as Microsoft's PowerPoint. To this end, geoscientists make screenshots or videos of the data visualized in PRo3D and embed them in the slide show. There are three main issues inherent to this approach, namely (1) a considerable effort required to create, edit, and embed artifacts like images and videos, (2) the static nature of presentations relying on prepared snippets of data, and (3) the difficulty of preserving the 3D context of geological data in screenshots. Our integrated solution overcomes these issues by letting the user create stories as slide shows directly within PRo3D. This reduces the effort required to create a story as no artifacts have to be created, edited, and imported into additional tools. Each slide is associated with a certain camera view, which can be changed interactively on the fly during the presentation. Consequently, the data can be presented in a dynamic way, which is useful in multiple scenarios. For example, in traditional live talks these interaction possibilities allow the presenter to show the data in a way that helps answering questions from the audience. For self-running presentation scenarios, the consumer may directly interact with the data, which can improve understanding in a similar fashion. Moreover, camera animations between individual slides and interaction with the actual data help to retain 3D spatial relations [BB99, WH07].

Another advantage of our solution over traditional workflows is the inclusion of a provenance-based model similar to CLUE [GLG+16]. Provenance information allows the geoscientist to include not only their results but also how they arrived at those results directly into their stories. To this end, interaction provenance is automatically captured as the user proceeds with their analyses, and represented as a tree in a dedicated window. The provenance tree gives an overview over the analysis session and also allows the user to revisit past states easily. Stories build upon provenance by associating each slide with a specific analysis state, i.e. a node in the provenance tree. As a result, users may switch

back from the presentation stage to the exploration stage at any point of a story. This facilitates understanding, verifying and building upon results of others, which is essential in an iterative scientific workflow [DC02, Ple18].

We demonstrated our prototype as a proof of concept that illustrates how our provenance-based storytelling approach works in practice. It is fully implemented in PRo3D and can be used to create sound geological stories. As a prototype, our implementation still lacks features that are required in a fully functional storytelling solution. For example, it is not possible to create slides containing formatted text, imported images or other media, which are useful to present introductory information. Moreover, some geological tools such as sedimentary logs are not supported, since PRo3D itself does not offer such functionality yet. Another limitation of our implementation is how provenance is visualized. Scalable graph visualization is beyond the scope of this thesis; therefore, we settled with a simple node-link diagram visualization for the provenance tree. For a more sophisticated solution, we refer to the excellent work by Gratzl et al. who use an LoD-based visualization technique [GLG+16]. Scalability is also reduced by how our solution captures and represents provenance. First of all, we employ a simple state-based approach for representing provenance information. Each node contains a full analysis state, resulting in a considerable memory overhead as the provenance tree grows. A more efficient solution would be to only track actions (i.e. differences between nodes), or to apply a hybrid of state-based and action-based representation [HMSA08]. As discussed in Section 4.2, another crucial aspect of provenance related to scalability is the level of abstraction at which it is represented. Ideally, a high level of abstraction resembling the actual analysis workflow is applied. Our solution uses a low-level approach due to a lack of feasible alternatives within the scope of this diploma thesis.

Future work will have to address these issues related to scalability. Foremost, the question of how the geoscientific workflow of outcrop interpretation can be represented as accurately as possible has to be answered. Is a high-level approach feasible? Is it even possible to capture insight provenance without heavily relying on the user to explicitly manage meta information?

In any case, we believe this work to be an important step in improving knowledge transfer in the domain of geological planetary exploration, and hope to motivate further research in this field.

Acronyms

Aardvark
An Advanced Rapid Development Visualization and Rendering Kernel 83, 84

AVOCADO
Adaptive Visualization of Comprehensive Analytical Data Origins 45

CLUE
Capture, Label, Understand, Explain 6, 27–31, 33, 37, 41, 57, 62, 69–71, 89

DoI Degree of Interest 30, 45, 62

DOM
Digital Outcrop Model 49–53, 56, 58, 59, 63, 64

DSL
domain-specific language 39

DTM
Digital Terrain Model 2

GUI
graphical user interface 19, 21, 22, 24, 26, 30, 31, 39, 41–45, 67, 83

HBDQ
Hanksville-Burpee Dinosaur Quarry 52, 73, 75–82

HiRISE
High Resolution Imaging Science Experiment 2

ICL Imperial College of London 73

LoD
level of detail 1, 31, 62, 63, 70, 90

MER
Mars Exploration Rover 1, 2, 50

MRI
Magnetic Resonance Imaging 21

MSL
Mars Science Laboratory 1

NASA
National Aeronautics and Space Administration 1, 9, 15, 18, 73

OPC
Ordered Point Cloud 1

OS operating system 34

PRo3D
Planetary Robotics 3D Viewer 1, 3–6, 33, 49–59, 61–65, 67–69, 71, 73, 74, 83–90

PRoViDE
Planetary Robotics Vision Data Exploitation 1

SVS
Scientific Visualization Studio 9, 15, 18

List of used geological terms

bed is a “layer of sediments or sedimentary rocks bounded above and below by more or less well-defined bedding surfaces. The smallest, formal lithostratigraphic unit of sedimentary rocks” [SWO98] 50, 54, 59

bedding plane
is a “planar or nearly planar bedding surface that visibly separates each successive layer of stratified sediment or rock (of the same or different lithology) from the preceding or following layer; a plane of deposition. It often marks a change in the circumstances of deposition, and may show a parting, a color difference, a change in particle size, or various combinations. A term commonly applied to any bedding surface even when conspicuously bent or deformed by folding” [SWO98] 3, 64, 93–95

bedrock
is “solid rock that underlies the soil and other unconsolidated material or that is exposed at the surface” [SWO98] 1, 94

cross-bedding
also known as cross-stratification, is “any layering in a sediment or sedimentary rock that is oriented at an angle to the depositional horizontal. These inclined strata most commonly form in sand and gravel by the migration of bedforms and may be preserved if there is net accumulation” [Nic09] 59

deposition
is the “laying down of any material by any agent such as wind, water, ice or by other natural processes” [SWO98] 25, 27, 93, 94

dip is the “maximum angle that a structural surface, (e.g. a bedding or fault plane) makes with the horizontal, measured perpendicular to the strike of the structure and in the vertical plane” [SWO98] 3, 55, 95, *see* strike

erosion
is the “process by which material weathered from rocks is transported by wind, water, ice, or abrasive solid particles, or by mass-wasting, as in rock falls and landslides” [Sch] 27

fault
is a “discrete surface (fracture) or zone of discrete surfaces separating two rock masses across which one mass has slid past the other” [SWO98] 3, 25–27, 93–95

fold is a “curve or bend of a planar structure such as rock strata, bedding planes, foliation, or cleavage” [SWO98] 25

lithologic
“pertaining to the physical character of a rock” [SWO98] 93

lithostratigraphy
is the “study and correlation of strata to elucidate Earth history on the basis of their lithology, or the nature of the well log response, mineral content, grain size, texture and color of rocks” [Sch] 93

outcrop
is the “part of a geologic formation or structure that appears at the surface of the earth” or an “actual exposure of bedrock at or above the ground surface” [SWO98] 1, 3–5, 49, 50, 52–54, 56, 57, 59, 63, 64, 67, 73, 90

paleocurrent
is an “ancient current (generally of water) that existed in the geologic past, whose direction is inferred from the sedimentary structures and textures of the rocks formed at that time” [Neu05] 52

sediment
is “material, both mineral and organic, that is in suspension, is being transported, or has been moved from its site of origin by water, wind, ice, or mass-wasting and has come to rest on the earth’s surface either above or below sea level. Sediment in a broad sense also includes materials precipitated from solution or emplaced by explosive volcanism, as well as organic remains; e.g., peat that has not been subject to appreciable transport” [Nic09] 25, 27, 50, 52, 54, 73, 74, 90, 93, 94

sedimentary rock
is a “consolidated deposit of clastic particles, chemical precipitates, or organic remains accumulated at or near the surface of the earth under "normal" low temperature and pressure conditions” [SWO98] 50, 93

sedimentology
is “the study of the processes of formation, transport and deposition of material that accumulates as sediment in continental and marine environments and eventually forms sedimentary rocks” [Nic09] 1, 73

slip is the "relative displacement of two formerly adjacent points that have been separated by faulting. Slip is used to describe motion along a fault with respect to the distance and direction that one side of the fault has moved relative to the other one. Slip is a vector, expressed in terms of distance and direction" [Sch] 26

stratified
"formed, arranged, or laid down in layers. The term refers to geologic deposits" [SWO98] 93

stratigraphy
is "the study of rocks to determine the order and timing of events in Earth history: it provides the time frame that allows us to interpret sedimentary rocks in terms of dynamic evolving environments" [Nic09] 1

strike
is the "compass direction or trend taken by a structural surface (e.g. a bed or fault plane) as it intersects the horizontal; used in combination with dip to describe the orientation of bedrock strata" [SWO98] 3, 55, 93, *see* dip

Bibliography

[Aara] Aardvark Media. `https://github.com/aardvark-platform/aardvark.media`. [Online; accessed June 14, 2019].

[Aarb] Aardvark Platform. `https://aardvark.graphics`. [Online; accessed August 24, 2019].

[AC12] A. Awasthi and S. S. Chauhan. A hybrid approach integrating affinity diagram, AHP and fuzzy TOPSIS for sustainable city logistics planning. *Applied Mathematical Modelling*, 36(2):573–584, 2012.

[ADV06] T. F. Ansary, M. Daoudi, and J.-P. Vandeborre. A Bayesian 3D search engine using adaptive views clustering. *IEEE Transactions on Multimedia*, 9(1):78–88, 2006.

[AGLW17] S. Almukhtar, T. Griggs, K. K. R. Lai, and T. Wallace. The islamic state: From insurgency to rogue state and back. `https://www.nytimes.com/interactive/2017/10/22/world/middleeast/isis-the-islamic-state-from-insurgency-to-rogue-state-and-back.html`, October 2017. [Online; accessed January 31, 2019].

[AHRL+15] F. Amini, N. Henry Riche, B. Lee, C. Hurter, and P. Irani. Understanding data videos: Looking at narrative visualization through the cinematography lens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1459–1468, New York, NY, USA, 2015. ACM.

[AWM10] H. Akiba, C. Wang, and K. L. Ma. AniViz: A template-based animation tool for volume visualization. *IEEE Computer Graphics and Applications*, 30(5):61–71, September 2010.

[BB99] B. B. Bederson and A. Boltman. Does animation help users build mental maps of spatial information? In *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, pages 28–35, Oct 1999.

[BCDH08] O. Biton, S. Cohen-Boulakia, S. B. Davidson, and C. S. Hara. Querying and managing provenance through user views in scientific workflows. In *2008 IEEE 24th International Conference on Data Engineering*, pages 1072–1081, April 2008.

[BGSJ07] C. Bond, A. Gibbs, Z. Shipton, and S. Jones. What do you think this is? "conceptual uncertainty" in geoscience interpretation. *GSA Today*, 17(11):4–10, November 2007.

[BGT+18] R. Barnes, S. Gupta, C. Traxler, T. Ortner, A. Bauer, G. Hesina, G. Paar, B. Huber, K. Juhart, L. Fritz, B. Nauschnegg, J.-P. Muller, and Y. Tao. Geological analysis of Martian rover-derived digital outcrop models using the 3D visualisation tool, Planetary Robotics 3D Viewer – PRo3D. *Earth and Space Science*, 2018.

[BH94] B. B. Bederson and J. D. Hollan. Pad++: a zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 17–26. ACM, 1994.

[BOH11] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec 2011.

[BYB+13] M. A. Borkin, C. S. Yeh, M. Boyd, P. Macko, K. Z. Gajos, M. Seltzer, and H. Pfister. Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2476–2485, 2013.

[CAB+14] L. Carata, S. Akoush, N. Balakrishnan, T. Bytheway, R. Sohan, M. Seltzer, and A. Hopper. A primer on provenance. *Commun. ACM*, 57(5):52–60, May 2014.

[CCM09] S. M. S. Cruz, M. L. M. Campos, and M. Mattoso. Towards a taxonomy of provenance in scientific workflow management systems. In *2009 Congress on Services - I*, pages 259–266, July 2009.

[Cza12] E. Czaplicki. Elm: Concurrent FRP for functional GUIs. *Senior thesis, Harvard University*, 2012.

[DC02] G. Dodig-Crnkovic. Scientific methods in computer science. In *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden, Skövde, Suecia*, pages 126–130, 2002.

[EKA+08] T. Ellkvist, D. Koop, E. W. Anderson, J. Freire, and C. Silva. Using provenance to support real-time collaborative design of workflows. In *International Provenance and Annotation Workshop*, pages 266–279. Springer, 2008.

[Ele] Electron. https://electronjs.org/. [Online; accessed August 27, 2019].

[Elm] The Elm Architecture. https://guide.elm-lang.org/architecture/. [Online; accessed June 14, 2019].

[Fig14] A. Figueiras. Narrative visualization: A case study of how to incorporate narrative elements in existing visualizations. In *Proc. 18th Int. Conf. Information Visualisation*, pages 46–52, July 2014.

[FKSS08] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science Engineering*, 10(3):11–21, May 2008.

[FSC+06] J. Freire, C. T. Silva, S. P. Callahan, E. Santos, C. E. Scheidegger, and H. T. Vo. Managing rapidly-evolving scientific workflows. In *International Provenance and Annotation Workshop*, pages 10–18. Springer, 2006.

[GAB+05] J. Grotzinger, R. Arvidson, J. Bell, W. Calvin, B. Clark, D. Fike, M. Golombek, R. Greeley, A. Haldemann, K. Herkenhoff, B. Jolliff, A. Knoll, M. Malin, S. McLennan, T. Parker, L. Soderblom, J. Sohl-Dickstein, S. Squyres, N. Tosca, and W. Watters. Stratigraphy and sedimentology of a dry to wet eolian depositional system, Burns formation, Meridiani Planum, Mars. *Earth and Planetary Science Letters*, 240(1):11–72, 2005. Sedimentary Geology at Meridiani Planum, Mars.

[Gal17] J. Gallier. *Discrete Mathematics, In Progress*. Philadelphia: Springer, 2017.

[GB05] T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290. ACM, 2005.

[GFI16] S. N. Goodman, D. Fanelli, and J. P. A. Ioannidis. What does research reproducibility mean? *Science Translational Medicine*, 8(341):341ps12–341ps12, 2016.

[GGL+15] S. Gratzl, N. Gehlenborg, A. Lex, H. Strobelt, C. Partl, and M. Streit. Caleydo web: An integrated visual analysis platform for biomedical data. In *Poster Compendium of the IEEE Conference on Information Visualization (InfoVis' 15)*. IEEE, 2015.

[GLG+16] S. Gratzl, A. Lex, N. Gehlenborg, N. Cosgrove, and M. Streit. From visual exploration to storytelling and back again. *Computer Graphics Forum*, 35(3):491–500, 2016.

[GP01] N. Gershon and W. Page. What storytelling can do for information visualization. *Commun. ACM*, 44(8):31–37, August 2001.

[GS06] D. P. Groth and K. Streefkerk. Provenance and annotation for visual exploration systems. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1500–1510, November 2006.

[GSK+14] J. P. Grotzinger, D. Y. Sumner, L. C. Kah, K. Stack, S. Gupta, L. Edgar, D. Rubin, K. Lewis, J. Schieber, N. Mangold, R. Milliken, P. G. Conrad, D. DesMarais, J. Farmer, K. Siebach, F. Calef, J. Hurowitz, S. M. McLennan, D. Ming, D. Vaniman, J. Crisp, A. Vasavada, K. S. Edgett, M. Malin, D. Blake, R. Gellert, P. Mahaffy, R. C. Wiens, S. Maurice, J. A. Grant, S. Wilson, R. C. Anderson, L. Beegle, R. Arvidson, B. Hallet, R. S. Sletten, M. Rice, J. Bell, J. Griffes, B. Ehlmann, R. B. Anderson, T. F. Bristow, W. E. Dietrich, G. Dromart, J. Eigenbrode, A. Fraeman, C. Hardgrove, K. Herkenhoff, L. Jandura, G. Kocurek, S. Lee, L. A. Leshin, R. Leveille, D. Limonadi, J. Maki, S. McCloskey, M. Meyer, M. Minitti, H. Newsom, D. Oehler, A. Okon, M. Palucis, T. Parker, S. Rowland, M. Schmidt, S. Squyres, A. Steele, E. Stolper, R. Summons, A. Treiman, R. Williams, A. Yingst, and M. S. Team. A habitable fluvio-lacustrine environment at Yellowknife Bay, Gale Crater, Mars. *Science*, 343(6169), 2014.

[GZ09] D. Gotz and M. X. Zhou. Characterizing users' visual analytic activity for insight provenance. *Information Visualization*, 8(1):42–55, 2009.

[HD11] J. Hullman and N. Diakopoulos. Visualization rhetoric: Framing effects in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2231–2240, Dec 2011.

[HDR+13] J. Hullman, S. Drucker, N. H. Riche, B. Lee, D. Fisher, and E. Adar. A deeper understanding of sequence in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2406–2415, December 2013.

[HGE+11] A. G. Hayes, J. P. Grotzinger, L. A. Edgar, S. W. Squyres, W. A. Watters, and J. Sohl-Dickstein. Reconstruction of eolian bed forms and paleocurrents from cross-bedded strata at Victoria Crater, Meridiani Planum, Mars. *Journal of Geophysical Research: Planets*, 116(E7), 2011.

[HGS+11] G. J. Hampson, M. R. Gani, K. E. Sharman, N. Irfan, and B. Bracken. Along-strike and down-dip variations in shallow-marine sequence stratigraphic architecture: Upper Cretaceous Star Point Sandstone, Wasatch Plateau, Central Utah, USA. *Journal of Sedimentary Research*, 81(3):159–184, 2011.

[HMSA08] 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, November 2008.

[HR07] J. Heer and G. Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, Nov 2007.

[HSMT15] G. Haaser, H. Steinlechner, S. Maierhofer, and R. F. Tobler. An incremental rendering VM. In *Proceedings of the 7th Conference on High-Performance Graphics*, HPG '15, pages 51–60, New York, NY, USA, 2015. ACM.

[JKMG02] T. J. Jankun-Kelly, K.-L. Ma, and M. Gertz. A model for the visualization exploration process. In *Proc. VIS 2002. IEEE Visualization*, pages 323–330, November 2002.

[KM13] R. Kosara and J. Mackinlay. Storytelling: The next step for visualization. *Computer*, 46(5):44–50, May 2013.

[KNF+01] S. R. Klemmer, M. W. Newman, R. Farrell, M. Bilezikjian, and J. A. Landay. The designers' outpost: A tangible interface for collaborative web site. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, pages 1–10, New York, NY, USA, 2001. ACM.

[KSJ+14] B. C. Kwon, F. Stoffel, D. Jäckle, B. Lee, and D. Keim. VisJockey: Enriching Data Stories through Orchestrated Interactive Visualization. In *Computation+Journalism Symposium 2014*, 2014.

[KTPG+02] S. R. Klemmer, M. Thomsen, E. Phelps-Goodman, R. Lee, and J. A. Landay. Where do web sites come from?: Capturing and interacting with design history. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 1–8, New York, NY, USA, 2002. ACM.

[LHV12] E. M. Lidal, H. Hauser, and I. Viola. Geological storytelling: Graphically exploring and communicating geological sketches. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 11–20, Goslar Germany, Germany, 2012. Eurographics Association.

[LLCF10] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. Prospective and retrospective provenance collection in scientific workflow environments. In *2010 IEEE International Conference on Services Computing*, pages 449–456. IEEE, 2010.

[LRIC15] B. Lee, N. H. Riche, P. Isenberg, and S. Carpendale. More than telling a story: Transforming data into visually shared stories. *IEEE Computer Graphics and Applications*, 35(5):84–90, September 2015.

[McC06] S. McCloud. *Making Comics: Storytelling Secrets of Comics, Manga, and Graphic Novels*. HarperCollins Publishers L.L.C., 2006.

[MLF+12] K. L. Ma, I. Liao, J. Frazier, H. Hauser, and H. N. Kostis. Scientific storytelling using visualization. *IEEE Computer Graphics and Applications*, 32(1):12–19, January 2012.

[MRS+13] E. Maguire, P. Rocca-Serra, S. Sansone, J. Davies, and M. Chen. Visual compression of workflow visualizations with automated detection of macro motifs. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2576–2585, Dec 2013.

[MS11] P. Macko and M. Seltzer. Provenance map orbiter: Interactive exploration of large provenance graphs. In *3rd USENIX Workshop on the Theory and Practice of Provenance*, pages 1–6, 2011.

[MSOI+02] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[MT14] N. Mahyar and M. Tory. Supporting communication and coordination in collaborative sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1633–1642, 2014.

[NCE+11] C. North, R. Chang, A. Endert, W. Dou, R. May, B. Pike, and G. Fink. Analytic provenance: Process+interaction+insight. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 33–36, New York, NY, USA, 2011. ACM.

[Neu05] K. K. E. Neuendorf. *Glossary of geology*. Springer Science & Business Media, 2005.

[Nic09] G. Nichols. *Sedimentology and stratigraphy*. John Wiley & Sons, 2009.

[Nor06] C. North. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, 26(3):6–9, May 2006.

[Par12] R. Parent. *Computer Animation: Algorithms and Techniques*. Newnes, 2012.

[PJ95] S. G. Parker and C. R. Johnson. SCIRun: A scientific programming environment for computational steering. In *Supercomputing '95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, pages 52–52, Dec 1995.

[Ple18] H. E. Plesser. Reproducibility vs. replicability: A brief history of a confused terminology. *Frontiers in Neuroinformatics*, 11:76, 2018.

[PMT+15] G. Paar, J.-P. Muller, Y. Tao, T. Pajdla, M. Giordano, E. Tasdelen, I. Karachevtseva, C. Traxler, G. Hesina, L. Tyler, R. Barnes, S. Gupta, and K. Willner. PRoViDE: Planetary Robotics Vision Data Processing and Fusion. In *European Planetary Science Congress*, volume 10, 2015.

[RBL+17] D. Ren, M. Brehmer, B. Lee, T. Höllerer, and E. K. Choe. ChartAccent: Annotation for data-driven storytelling. In *Proc. IEEE Pacific Visualization Symp. (PacificVis)*, pages 230–239, April 2017.

[RESC16] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing provenance in visualization and data analysis: An organizational framework of provenance types and purposes. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):31–40, January 2016.

[RT81] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, SE-7(2):223–228, March 1981.

[Sch] Schlumberger. The Oilfield Glossary. `https://www.glossary.oilfield.slb.com`. [Online; accessed February 15, 2019].

[SGP+19] H. Stitz, S. Gratzl, H. Piringer, T. Zichner, and M. Streit. KnowledgePearls: Provenance-based visualization retrieval. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):120–130, Jan 2019.

[SH10] E. Segel and J. Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, November 2010.

[SKS+08] C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, 2008.

[SLSG16] H. Stitz, S. Luger, M. Streit, and N. Gehlenborg. AVOCADO: Visualization of workflow–derived data provenance for reproducible biomedical research. *Computer Graphics Forum*, 35(3):481–490, 2016.

[SPG05] Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, September 2005.

[SVK+07] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, Nov 2007.

[SvW08] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1237–1246. ACM, 2008.

[SWO98] P. J. Schoeneberger, D. A. Wysocki, and C. G. Olson. Glossary of landform and geologic terms. *The National Soil Survey Handbook*, September 1998.

[TB03] K. Thompson and D. Bordwell. *Film history: An introduction*, volume 205. McGraw-Hill Boston, 2003.

[TC06] J. J. Thomas and K. A. Cook. A visual analytics agenda. *IEEE Computer Graphics and Applications*, 26(1):10–13, Jan 2006.

[The18] The Economist. All politics is identity politics: How to forecast an American's vote. `https://www.economist.com/graphic-detail/2018/11/03/how-to-forecast-an-americans-vote`, November 2018. [Online; accessed January 31, 2019].

[TKE12] R. M. Tarawaneh, P. Keller, and A. Ebert. A general introduction to graph visualization techniques. In *Visualization of Large and Unstructured Data Sets: Applications in Geospatial Planning, Modeling and Engineering-Proceedings of IRTG 1131 Workshop 2011*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2012.

[TRL+18] C. Tong, R. C. Roberts, R. S. Laramee, K. Wegba, A. Lu, Y. Wang, H. Qu, Q. Luo, and X. Ma. Storytelling and visualization: A survey. In *VISIGRAPP*, 2018.

[TSS+18] M. Thöny, R. Schnürer, R. Sieber, L. Hurni, and R. Pajarola. Storytelling in interactive 3d geographic visualization systems. *ISPRS International Journal of Geo-Information*, 7(3), 2018.

[WH07] M. Wohlfart and H. Hauser. Story telling for presentation in volume visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07, pages 91–98, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.