A Layered Approach to Lightweight Toolchaining in Visual Analytics

Hans-Jörg Schulz[1][0000−0001−9974−535X], Martin Röhlig[2], Lars Nonnemann[2][0000−0002−4156−2179], Marius Hogräfer[1][0000−0002−3649−9339], Mario Aehnelt[3], Bodo Urban[2][0000−0003−4968−4506], and Heidrun Schumann[2]

[1] Department of Computer Science, Aarhus University, Denmark
[2] Institute of Visual and Analytic Computing, University of Rostock, Germany
[3] Fraunhofer Institute for Computer Graphics Research, Rostock, Germany

Abstract. The ongoing proliferation and differentiation of Visual Analytics to various application domains and usage scenarios has also resulted in a fragmentation of the software landscape for data analysis. Highly specialized tools are available that focus on one particular analysis task in one particular application domain. The interoperability of these tools, which are often research prototypes without support or proper documentation, is hardly ever considered outside of the toolset they were originally intended to work with. To nevertheless use and reuse them in other settings and together with other tools, so as to realize novel analysis procedures by using them in concert, we propose an approach for loosely coupling individual visual analytics tools together into toolchains. Our approach differs from existing such mechanisms by being lightweight in realizing a pairwise coupling between tools without a central broker, and by being layered into different aspects of such a coupling: the usage flow, the data flow, and the control flow. We present a model of this approach and showcase its usefulness with three different usage examples, each focusing on one of the layers.

Keywords: Visual Analytics · Software Integration · View Coordination.

1 Introduction

Visual Analytics (VA) encompasses the frequent and fluent back and forth between computational analyses and interactive visual exploration. This process can get quite involved, with many specialized analysis steps from data wrangling [16], data preprocessing [18] and data exploration [6], all the way to model building [10] and investigating uncertainties [36]. Each of these specialized analysis steps comes with its own methods and tools that may or may not be used, depending on the analysis objectives known up front and on the findings made during an analysis. Such a flexible orchestration and use of VA tools at the analyst's direction is hard to capture with preconfigured analysis pipelines as they are commonly employed in data science. This observation has led so far as

some researchers suggesting to recoin the common phrase of "the human *in* the loop" into "the human *is* the loop" [5]. Yet in order to put the analyst at the helm of the analysis process and to be able to reconfigure and reparametrize the concerted use of multiple VA tools, a mechanism is needed to accomplish that.

We follow this line of thought and propose the concept of *layered VA toolchains* that, unlike fixed pipelines realized within an integrated VA framework, allow for a more flexible coupling of independent VA tools. Through this coupling, the otherwise autonomous tools form loose multi-tool ensembles that coordinate aspects of their joint use. This is achieved by differentiating VA tool coupling into three separate concerns:

- *Carrying out one tool after another*: The most fundamental characteristic of a concerted use of VA tools is the selection of suitable tools, as well the possible sequences in which they can be used. Capturing this first essential aspect allows guiding a user through a toolchain by automatically providing the user the right tool at the right time.
- *Funneling data from one tool into the next*: VA tools require an input to produce an output. If one tool's input is the resulting output from a preceding tool, this needs to be passed and possibly transformed between them. Capturing this aspect allows the toolchain to handle this process, automatically providing the user with the right data at the right time.
- *Coordinating functional aspects between tools*: Beyond the data, VA tools may require to set parameters or adjust the methods employed to produce their outcome. Capturing these functional characteristics allows keeping them consistent between tools, automatically providing the right parameters and presets at the right time to the user.

In this paper, which constitutes an extension of our previous work [35], we detail, discuss, and exemplify this threefold separation of concerns into layers for VA toolchaining. After looking briefly at related work for coordinating tools and views in Sec. 2, we present our conceptual model that captures these three layers in Sec. 3. We then discuss the benefits for the user that each individual layer yields and give concrete examples for them in Sec. 4 through Sec. 6.

2 Related Work

Prior research on tool coordination has resulted in a variety of frameworks such as *OBVIOUS* [7] or *VisMashup* [31]. They provide an interoperability layer on code level, which offers the necessary functionality to programmers and developers for coupling their VA tool with other tools utilizing the same framework. Coordination without code access relies usually on the exchange of data, views, or both, as they are produced by the VA tools.

Coordination among visualization tools through data exchange dates back to 1997 (cf. Visage [17]). Most approaches for data level coordination rely on a centralized mechanism. On the one hand, this can be a central database that

acts as model in a model-view-controller (MVC) mechanism. Examples are *Snaptogether* [22] and EdiFlow [2] that both use a relational database as underlying central data storage for a set of tools. On the other hand, coordination can employ a communication bus or service bus to broker messages among tools, as for example done by the *Metadata Mapper* [28]. In case no centralized data exchange mechanism is provided, tools use custom connectors to pass data. For closed source tools, this may rely on *screen poking* and *screen scraping* [8, 14].

Coordination among visualization tools based on the views they produce can be achieved by exchanging and combining user interface (UI) components and graphical outputs from different systems, as it is proposed by approaches, such as *ManyVis* [30]. As the coordination of applications through the exchange of data or views are independent, they can be combined as necessary to achieve the required flexibility for a given analysis. One of the few examples for coordination using both, data and views, is *Webcharts* [9]. To make the interoperability of the tools available to the user, standard visualization frameworks usually display such an interlinked setup in multiple coordinated views [4, 27] or fused visualizations [21]. In some cases, the toolchain itself is visually encoded in a (directed) graph layout that shows its connections [40, 11]. When arranging UIs side-by-side is not sufficient, a common interface may be glued together from individual interface parts. WinCuts [39] and Façades [38] enable users to do so by cutting out pieces of multiple UIs and interactively composing them into one tailored UI. For web-based ensembles, existing UIs are combined in mashups [23].

In most application domains from climatology to biomedicine, the current practice in data analysis is to simply use independent VA tools one after the other. And as these domains also feature a diverse set of file formats and data conventions, it is often already considered a high level of interoperability when the different tools can work on the same data files, effectively easing the relay of results from one tool into the next. This current data analysis practice forms the basis of our coordination approach, which is described in the following.

3 A Layered Approach to Lightweight Toolchaining

As exemplified by the previous section, prior work on coordinating and linking visualization and analysis tools has focused mainly on the implementation and software engineering aspects of such mechanisms – i.e., the *algorithmic level* in terms of Munzner's four levels of visualization design [20, ch.4.3]. Our toolchaining approach puts its focus on the remaining three levels in the nested visualization design model that remain so far mostly undiscussed and unexplored:

– The *situational level* that aims to understand the domain situation – i.e., the application domain, the application data and questions, as well as the analytical processes that derive from them. In the context of toolchaining, the information captured on this level is about the used toolset in a domain, as well as common orders of use for domain-specific analyses. From this perspective, the situational level defines the *usage flow* among tools.

- The *abstraction level* that aims to understand the data and what is done with it in a domain-independent way. In the context of toolchaining, this captures the exchange of data between tools – i.e., which data in which data format as a result of which data transformation carried out by a VA tool. Consequently, this level defines foremost the *data flow* among tools.
- The *encoding and interaction level* that aims to understand how the data is visually displayed and how this display can be adapted. In the context of VA tools, this translates into parameter settings that govern the display, that can be changed to adapt the display, and that can be passed along for consistency of displays among tools. Hence, from the perspective of toolchains, this level defines the *control flow*.

3.1 The Principal Ideas behind the Approach

Our approach is different from established coordination approaches in two ways: its layered structure for better separation of concerns among different notions of tool coordination, and its lightweight bottom-up realization for introducing automation to preexisting manual toolchains in an incremental and minimally invasive manner.

Our approach for toolchaining is *layered* in the sense that each of the individual levels of usage flow, data flow, and control flow can be used to affect a coordination among tools – either by themselves and with respect to the particular aspect of toolchaining they capture, or in combination with each other. For example, two tools may only be coordinated with regard to their data flow. This means that users still have to invoke one tool after the other and parametrize them manually, but that the data exchange between them is aided by a toolchaining mechanism, which can range from injecting a tool for handling format conversion to fully automated, bidirectional live data exchange. It is also possible that two tools are coordinated with regard to more than one level – e.g., their usage flow and their control flow. This could mean that after closing one tool, the next tool in the usage flow is automatically started together with a properly parametrized view that reflects any manual fine-tuning done in previously used tools, but that the data output and input still needs to be carried out manually.

Our approach for toolchaining is *lightweight* in the sense that we consider its realization to be pairwise between tools. This means, instead of having to provide an all-encompassing framework that coordinates between all possible VA tools and anticipates all possible combinations in a top-down manner, we instead use a bottom-up approach that works by coordinating only between those tools and on those levels, where this makes most sense. This approach allows us to utilize different coordination mechanisms between tools, depending on which interfaces they offer. As a result, coordination between tools in a toolchain can be introduced in a step-wise manner bridging the most tedious gaps that incur the most manual labor first, thus adding coordination and automation where it is needed most.

These ideas are directly reflected in the following coordination model that captures this vision of a layered, pairwise toolchaining.

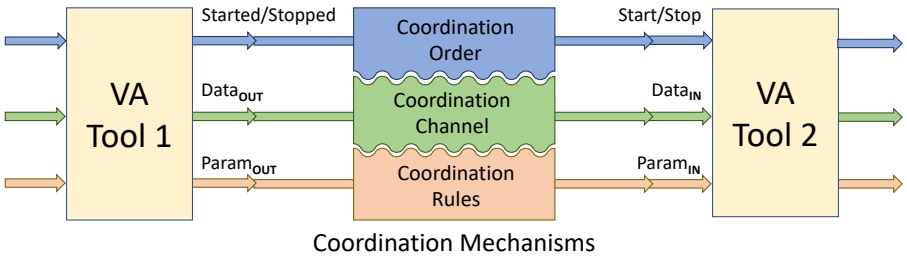


Fig. 1. Conceptual abstraction of lightweight coordination between two VA tools. The *coordination order* models any temporal dependency between two tools (i.e., their subsequent or concurrent use). The *coordination channels* capture ways to exchange data between tools, including any necessary data transformations along the way. The *coordination rules* describe automated syncing of interactive controls between tools by means of exchanging their associated parameters.

3.2 A Model for Layered Lightweight Toolchaining

To characterize coordination among VA tools on the three levels of usage flow, data flow, and control flow, we need adequate means to describe coordination in such a faceted way. In line with common view coordination models from the literature – e.g., [43] and [3], we propose to model VA tool coordination as a graph. In this graph, VA tools constitute the nodes and directed edges capture the usage flow, data flow, and control flow between pairs of VA tools. We outline these parts of our coordination model, as well as how to establish and utilize them in the following.

Modeling the VA Tools. VA tools form the natural basis of our coordination model, in the same way as they form the basis of the analytic toolchain. As these tools can come in any shape or form – from closed source to open source software, from simple command line tools to full-fledged VA frameworks – we follow the principle of making no prior assumptions about these tools and consider them as black boxes that are characterized by their inputs and outputs on the three levels introduced above. Fig. 1 shows two tools modeled in this way, capturing the following I/O possibilities as *ports* of a VA tool:

Start(ed)/Stop(ped): Starting and stopping a tool, as well as being notified when it has started or stopped is the most fundamental of all tool capabilities. We model the invocation and termination of a tool via the port *Start*/*Stop* and the respective notifications via the port *Started*/*Stopped*.

Data (in/out): A VA tool requires input data on which to perform the analysis. The passed input may yield results that the tools passes back in return. It is common for most VA tools to have some ways of loading data, which we model as a port $Data_{IN}$ and saving results via the port $Data_{OUT}$. Note that the term data subsumes numerical data as well as image data.

Parameters (in/out): A VA tool's behavior is usually parametrizable. If parameters can be passed to the tool – e.g., as command line options when invoking it – this is modeled via the port $Param_{IN}$. If parameter settings can also be stored away for later re-use – e.g., as a config-file – this can be captured through the port $Param_{OUT}$

Note that these ports are abstractions that we use to model the respective possibility to manage an independent VA tool, regardless of whether these are indeed provided by the tool itself or by some other entity, like the operating system. For example, invoking a tool and observing its state is on a technical level rarely done via the tool itself, but instead utilizes the operating system's process and window managers.

It is further noteworthy, that a VA tool does not necessarily possess all of these I/O possibilities. In some cases, we may be able to work around them – for example, when inferring otherwise inaccessible parameters from a tool (i.e., $Param_{OUT}$) from the updated results (i.e., $Data_{OUT}$), e.g. by extracting color scales from visualizations [24]. In other cases, we might need the user to step in – for example, by manually setting parameters or choosing among presets.

Modeling the Usage Flow. At the lowest level of coordination, there is the temporal order of VA tools – i.e., the usage flow determined by the analysis scenario and its domain. The usage flow captures the toolchain as a succession of VA tools, as these are used by the user in pursuit of a particular analysis goal. It can include subsequently used tools, as well as concurrently used tools. Tool coordination modeled and realized at this level is able to bring up the right tools at the right time, as it is foreseen by the usage flow.

In our model, this type of coordination is achieved through a bilateral connection among two tools indicating their *coordination order* – shown at the top in Fig. 1. This order basically starts one VA tool subsequently or concurrently, depending on another VA tool having been Started/Stopped:

- Subsequently: $(Tool_1.Stopped) \Longrightarrow (Tool_2.Start)$
- Concurrently: $(Tool_1.Started) \Longrightarrow (Tool_2.Start)$

Coordination on this level already provides as much as an automated guidance of the user along a predefined path of analysis, in the spirit of approaches such as *Stack'n'Flip* [37]. While the users still have to do everything else themselves – such as moving data back and forth between the tools, or parametrizing their visualizations to match up – the coordination order between tools allows to automatically move the VA toolchain forward as the interactive analysis progresses. Besides providing convenience for the user, this also ensures comparability between different analysis runs as VA tools are always invoked in line with the predefined usage flow. This is important in cases where carrying out analysis steps in different orders yields different results – e.g., when performing dimension reduction and clustering. But it is also useful to ensure that no VA tool is left out by mistake – e.g., forgetting to normalize the data before processing it.

Modeling the Data Flow. The next level of coordination is about getting data in and out of a VA tool. Besides starting and stopping a tool, this is arguably the most important aspect of a VA tool: without data, there is nothing to work with. The data flow captures how input data is passed into VA tools, transformed into analysis results and passed on to the next VA tool as input again. At this level, VA tool coordination automatically hands off data from tool to tool as the user proceeds with the analysis along the toolchain – i.e., the flow of data tends to follow the usage flow. Depending on where in the visualization pipeline a VA tool operates, data can refer to raw data, prepared data, focus data, geometric data, or image data [32, 41].

This coordination is achieved via *coordination channels* – shown in the middle of Fig. 1. These capture the data exchange from a VA tool's output to another's input, as well as any data transformations f_D along the way:

$$Tool_1.Data_{OUT} \xRightarrow{f_D} Tool_2.Data_{IN}$$

Coordination on this level automatically delivers the right data to the right tools at the right time, very much like data flow-oriented visualization frameworks do. While the user still has to manage what is to happen with that data in a currently used VA tool – i.e., interactively parametrizing computations and visualizations – having input data and results from previously used VA tools delivered automatically to the current VA tool takes care of any tedious conversion between different data formats, competing standards, or sometimes just different versions or interpretations of the same standard [34]. This automation also ensures that the user is not working on stale data, as the coordination channels always deliver the most current data and manually loading an old dataset can only happen on purpose, not by accident. Coordination channels can further be used to store a snapshot of the last data passed by them, so that when revisiting a previously used VA tool its last input data is still available, thus providing a coherent analysis experience forward *and* backward along the toolchain.

Note that "passing data" can be a complex problem and research on data integration and scientific workflows has established various approaches to do so [19]. For modeling the coordination on data level, it is enough to know that one of these approaches underlies a channel and realizes the data transport.

Modeling the Control Flow. The last level of coordination is about having interactively set or changed controls in one VA tool also reflected in other tools by exchanging their associated parameters. This is probably the most common understanding of coordination, where for example, filter operations, selections, or adaptations of a color scale in one tool will also affect other tools. Though, how exactly these will affect subsequently or concurrently used tools is subject to the concrete nature of the coordination. Our model captures the settings and interactive choices made within tools as expressed through their parameters – e.g., numerical filter criteria, color scales, or transformation functions that can be tuned in one tool and reused in another. At this level, VA tool coordination automates the synchronization of interactive changes made across tools, which

mostly relates to the visualization or UI of these tools as these are the common means with which the user interacts.

In our model, this coordination is achieved via *coordination rules* – shown at the bottom in Fig. 1. These rules capture not only the mere passing of parameters between tools, but also the "coordination logic" behind the linking they realize. For example, in cases where this linking is supposed to have changes made in one tool being reflected likewise in another tool – e.g., a selection – the coordination rule relays the corresponding parameters as they are. Yet in cases, were this linking is supposed to have a complementing effect – e.g., the other tool not showing the selection, but everything that was not selected – the coordination rule relays the inverse of the corresponding parameters. Coordination rules are modeled as functions f_P with the identity $f_P = id$ as a default:

$$Tool_1.Param_{OUT} \xRightarrow{f_P} Tool_2.Param_{IN}$$

Coordination on this level can be used to provide any of the common coordination patterns, such as linking & brushing, navigational slaving, or visual linking [42] – provided that the corresponding view and data parameters are captured and accessible to be passed between tools. As it was the case for coordination channels, the user still has to manage and steer the currently used VA tool, but where applicable, this steering is picked up on and automatically mirrored or complemented in other VA tools. This includes not only tools that are used concurrently, but also those used subsequently, as interactive selections or carefully tuned color scales can be passed on as parameters to the next tool in the toolchain as well. This relieves the users from having to make the same interactive adjustments multiple times for each VA tool they are working with, and it guarantees consistent settings across tools. This is helpful when trying to compare results or for tracking certain data items across the toolchain.

3.3 Combining Tools and Flows

Taken together, we yield a 3-tiered model of lightweight coordination among VA tools comprised of the VA tools, as well as of the three coordination levels to realize different aspects of coupling among those tools. It can thus be expressed as a 4-tuple $CM = (\mathit{Tools}, \mathit{UsageFlow}, \mathit{DataFlow}, \mathit{ControlFlow})$ consisting of the following sets:

- The set of *Tools* as it is given by the analysis setup.
- The *UsageFlow*, defined as the set of all coordination orders (*Source*, [*Started*|*Stopped*], *Target*, [*Start*|*Stop*]) capturing the execution sequence between a source and a target tool.
- The *DataFlow*, defined as the set of all coordination channels (*Source*, *Target*, $f_D : Data_{OUT} \mapsto Data_{IN}$) capturing data transfer and transformation (f_D) between a source and a target tool.
- The *ControlFlow*, defined as the set of all coordination rules (*Source*, *Target*, $f_P : Param_{OUT} \mapsto Param_{IN}$) capturing parameter exchange and modification (f_P) between a source and a target tool.

All three sets of *UsageFlow*, *DataFlow*, and *ControlFlow* model the pairwise coordination between VA tools: The *UsageFlow* contains the answer to the question of which parts of the toolchain to automate tool-wise and in which order. The *DataFlow* comprises the answer to the question along which parts of the toolchain to transmit data using which data transformation. And the *ControlFlow* captures the answer to the question of which coordination logic is realized between tools of the chain. The three sets have in common that they describe the coupling between pairs of tools, so that each of these couplings can likewise be understood as sets of directed edges that taken together define a graph topology over the set of VA tools. They thus capture coordination in a bottom-up fashion by coupling two tools at a time and then combining these pairwise couplings into larger coordination mechanisms. This combination is for example done by adding reciprocity (combining the unidirectional coupling between two tools A and B, and between B and A) or by employing transitivity (coupling two tools A and B through an intermediary tool C).

The strength of this model is its decentralized, pairwise structure that requires neither a central broker or mediator, nor does it force any architectural changes on the used VA tools. Instead of trying to lift a whole toolchain up to conform to a state-of-the-art coordination framework, we coordinate directly between two VA tools. This way, we can capture the full variety of different modes, degrees, and directions of coordination between different tools, instead of boiling the coordination down to the least common denominator among all involved tools. This directly benefits:

- the VA experts, who can introduce coordination incrementally – adding one level of coordination among one pair of tools at a time – and thus adaptively expand and refine coordination as it becomes necessary,
- the software experts, who can leverage whichever features a tool already provides to connect to it,
- and the end users, who can bring in additional tools, which may or may not already have couplings to the tools from a current toolchain.

The decentralized model is mainly targeted at the VA and software experts. While the end users are also thought to benefit from the provided flexibility, overlooking and taming such a "zoo of tools" can also become quite a challenge. To aid in doing so, our approach complements the decentralized coordination mechanism with interface arrangements that provide structure on top of the tool ensemble. These interface arrangements are described in the following section.

3.4 Interface Ensembles

When handling multiple VA tools, the mere orchestration of their application windows becomes a management task by itself that requires attention, effort, and time which would be better spent on the visual analysis itself. In order to reduce this overhead, we propose the notion of *interface ensembles*. These provide an organized view on the otherwise often overwhelming mesh of any number of

invisible pairwise coordination mechanisms springing to life depending on the currently used tools and the user's actions.

Interface ensembles consist first and foremost of a centralized panel for global views and interaction elements that concern the toolchain as a whole. We call this centralized panel the *Control Interface* and it serves at its core as a place in which to display information that cuts across tools and where to affect global changes and adjustments. In this function, the control interface can be utilized by all three levels of coordination:

- for the *Usage Flow*, we can use the control interface for example to display progress information and to invoke additional tools or choose between alternative analysis paths;
- for the *Data Flow*, we can use the control interface for example to archive snapshots of interesting intermediate results for later in-depth investigation along the lines of a bookmarking or multi clipboard system;
- for the *Control Flow*, we can use the control interface for example to display global information like legends and to set global parameters like color scales or which data field to use for labeling data items.

In addition to the control interface, interface ensembles employ structured ways of displaying UIs, so that they appear in a predictable manner that suits the current usage flow. Common ways of doing so are outlined in the following.

Individual VA Tool Use: The Tabbed UI. The usage flow among VA tools is a path: first a tool A is used, then a tool B, followed by a tool C, and so forth. This individual, subsequential use of VA tools as predefined by the coordination orders that connect the tools in a temporal sense, results in an exclusive use of a single UI at each point in time as shown in Fig. 2(a). What reads like an oversimplification at first is actually the most prevalent usage pattern in practice, as visual analysis is for the most part conducted as a linear series of very specific analysis steps, each carried out with a highly specialized analysis tool or view.

To the user, these tool sequences can be offered in a variety of ways. One way of displaying such sequential procedures is through a tabbed interface that opens each tool in a dedicated tab, with the tabs being ordered according to the tool sequence. This way, by clicking on the tabs, the user is always able to go back in the toolchain and to readjust some property in an earlier used tool – for example, manually moving a data item from one cluster into another one. Given that all other parameters and choices along the toolchain stay the same, these changes can be passed automatically through the appropriate channels and be processed by the appropriate rules to auto-update the current tool and its view. The tabbed interface can further be augmented by a wizard-like guidance that leads the user tab by tab along the path defined by the coordination orders.

Combined VA Tool Use: The Tiled Display. Sometimes, it makes sense to use VA tools not just one tool at a time, but to have access to subsequent tools of the toolchain at once. This can be the case, for example, when a data

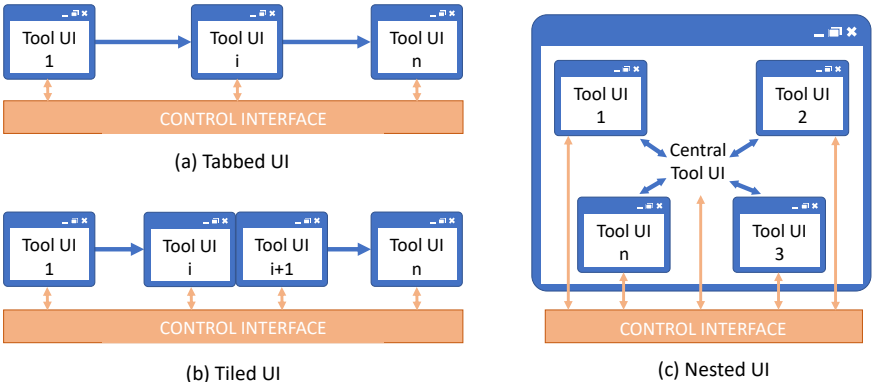


Fig. 2. Common UI layouts when dealing with multiple VA tools.

selection from one tool will serve as an input to the next tool and one needs to go back and forth between the two tools to try out and observe the effects of different selections. The topology is still a path topology, as shown in Fig. 2(b), but with two UIs being displayed at once to facilitate such combined use.

To the user, such setups are usually offered by tiling the display and showing the tools side by side, or by distributing them among multiple monitors. In this way, the tools are present on the screen at the same time to work with them without having to switch – i.e., sending one to the background and bringing another one to the front, as it would be the case in the tabbed interface. Synchronization features, such as linking & brushing and displaying visual links are desirable to make the back and forth between the tools more fluent.

Flexible VA Tool Use: The Nested UI. If VA tools are used more flexibly than a mere back and forth along a path of sequential tools, the resulting topology also gets more involved. A powerful example for this case is the star-shaped topology that is shown in Fig. 2(c) where all analysis steps start from a hub application or central VA tool. Such topologies support more complex usage flows that meander between multiple tools until their combined use yields an analysis result. This is often the case in comparative analyses where multiple windows and tools are needed to process, show, and relate different data subsets or different analytical procedures to each other.

To the user, the central tool is usually offered as an omnipresent overview of the data that is shown in a fashion similar to a background image. In this overview, users can select regions of interest into which to dive deeper by opening them up in other VA tools. The opened tools are shown as nested or superimposed views right in place where the selection was made. Making multiple selections opens multiple tools, effectively realizing the star-shaped topology. For this to work even with a dozen tools all scattered across the overview of the central VA tool, the overview/background needs a map-like appearance that serves well as a context for all the other UIs and makes their spatial relation meaningful.

The following sections will give a few first impressions of how this model for lightweight tool coordination can be employed in real-world analysis scenarios. To do so, each of the three sections focuses on the coordination of one particular layer and for one particular use case, capturing and utilizing the usage flow in Sec. 4, the data flow in Sec. 5, and the control flow in Sec. 6.

4 Providing the Right Tool at the Right Time

At the first layer of VA tool coupling, we define a temporal toolchain according to existing analysis procedures in a domain. This includes selecting suitable tools for the given tasks, determining sequences in which they should be used, and guiding a user through these sequences by automatically activating the right tool at the right time. We demonstrate the benefits of this layer with an editor to specify temporal chains of VA tools and its application in ophthalmology.

4.1 An Editor for Building a Temporal Toolchain

The first step of coupling independent tools is to put them in the right order. In many cases, numerous possible orders of tools are available. For example, in a larger analysis procedure, the order between two or more tools may be predefined, e.g., due to semantic constraints, while at the same time alternative execution sequences for the remaining tools exist. Thus, the first important question is: *who defines the order of tools?*

We argue that an appropriate answer to this question is to let the domain experts create the order of tools. That is because domain experts generally have the necessary context knowledge about which individual tool needs to be used when in their analysis procedure. However, this requires a simple interface for modeling a temporal toolchain that can be understood by non-technical users. This brings us to the second important question: *how can we support domain experts in defining orders of tools?*

As with every other temporal sequence, there is some ordering implied by an analytical toolchain. We choose to represent the orders of independent tools as directed acyclic graphs that contain multiple time steps with at least a single tool and optional data inputs necessary for carrying out the tools. In addition, we provide an editor with a simple user interface in order to support domain experts in creating such graphs. This requires us to 1) allow access to all tools that are needed, 2) allow access to all data that are needed to run these tools, and 3) allow to define connections between tools and data sources in the ordering process.

The main idea of our editor is to build a metaphorical bridge between the domain expert and the internal tool coupling processes. Hence, we choose an interface design that is similar to common presentation software, where a user can configure a sequence of slides to be shown in a later talk.

The display of our editor's user interface is divided into three containers that hold different types of information for the ordering process (Fig. 3):

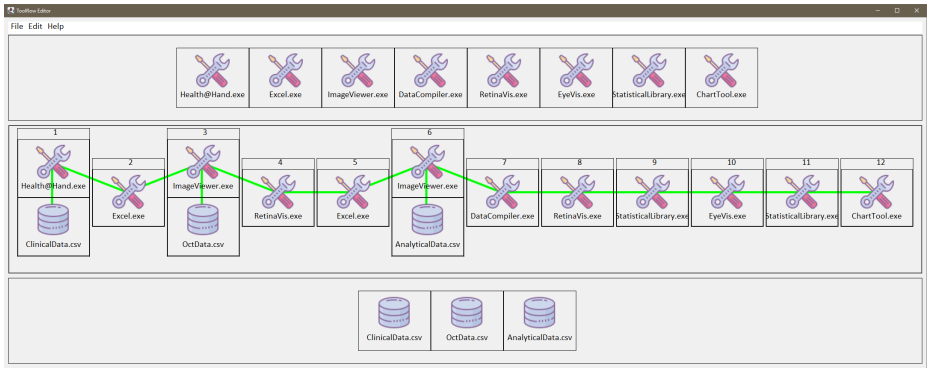


Fig. 3. Analytical process editor with temporal order of multiple tools used in an ophthalmic analysis process. The three panels in the editor's user interface represent the tool container (top), the graph container (middle), and the source container (bottom).

1. The **Tool Container** holds a collection of all executable tools. Imported tools can be dragged from this container and dropped into the graph container to either create new time steps or add tools to existing time steps.
2. The **Source Container** holds a collection of data sources. These data sources can be included in the graph to supply tools at a certain time step with input or supplementary data.
3. The **Graph Container** is the main part of the editor's user interface. It provides a space for the creation and modification of directed acyclic graphs. Each tool or data source can be dragged & dropped from the other containers into the graph container to order the analytical process. The resulting temporal toolchain can then be executed or saved as a JSON file by interacting with the main menu at the top of the user interface.

All necessary tools and data sources are imported into the editor by dragging them onto the two respective containers in the user interface. In fact, all of our import and export mechanisms are based on drag & drop interactions to facilitate an intuitive user experience. This is to enable non-technical users to adapt more easily and quickly to the temporal ordering process of independent tools.

All in all, our editor provides an easy-to-use interface that abstracts the temporal coupling of tools similar to common presentation software. At the same time, the editor is highly flexible with respect to the number of imported tools or data sources as well as to the size and complexity of created toolchains.

4.2 A Temporal Toolchain in Retinal Data Analysis

We applied our editor to build a tool sequence on top of a scientific analysis procedure for retinal data in ophthalmic research. For this purpose, we collaborated with a group of ophthalmic experts. We explained our objectives, discussed suitable analysis scenarios, and jointly identified challenges with their current

use of independent software tools without any kind of automated support for activating tools as needed.

Together, we defined the content of the tool container, data source container, and graph container of the editor for our use case in three steps. First, we learned which tools and data are required by taking a closer look at the retinal data analysis. Second, we temporally connected the tools by tracing the order in which they were applied in the experts' analysis procedure. Third, we assessed the benefits of this first layer of tool coupling by gathering informal feedback and reflecting on results.

Determining tools and data: The ophthalmic experts were interested in studying retinal changes of patients in relation to healthy controls. They started by gathering retinal 3D image data acquired via optical coherence tomography (OCT) and other clinical parameters of the two study groups from a clinical data management system. The data analysis was then carried out in three main steps:

1. Data preparation: Compilation of study groups and data quality checks.
2. Data exploration: Discovering differences between study groups and investigating relationships with clinical parameters.
3. Data presentation: Summarizing findings and reporting study results.

In this process, a set of eight diverse tools was applied. This included commercial software, e.g., device-specific OCT software and spreadsheet software, as well as statistics software and VA software, e.g., the R software environment [26] and a custom VA framework for retinal OCT data [29, 25]. The data required to carry out the tools comprised OCT scans and electronic health records for both patients and controls. All tools and datasets were imported into the tool container and the data source container of the editor, respectively.

Temporally connecting tools: Next, we focused on making the tools available when needed by automating their activation. We utilized our editor to model the temporal order of the tools as a directed graph according to an established analysis procedure currently in use in ophthalmic research. We started by sketching an initial tool sequence in the editor together with the experts. The visual presentation of temporal connections in the editor's user interface helped us to get an idea of when which tool was applied. Using the editor's direct drag & drop manipulation support, we were able to refine the initial sequence based on the experts' feedback and devise alternatives. Figure 3 shows the resulting sequence of applied tools in the user interface of the editor. With the editor's graph container at hand, the experts were able to create, refine, and store modeled tool sequences. They were also able to perform test runs via the editor to make sure the modeled tool sequences exactly matched their requirements.

Benefits of temporal tool coupling: Our observations and the ophthalmic experts' feedback are summarized with respect to the utility of the editor and supporting a temporal coupling of tools in general. Regarding the editor, we were able to

successfully model a tool sequence on top of the analysis procedure explained by the ophthalmic experts. In this regard, the editor proved to be a useful aid for this kind of work. In fact, other ophthalmic analysis procedures similar to the described use case were identified, including studying specific diseases or investigating corneal study data. While the general analysis approach and involved tasks are comparable, the required tools and the order in which they are applied may differ. In this context, our editor supports defining new temporal tool sequences and helps to revisit and adapt existing ones. The latter becomes necessary if medical devices and respective software are changed in a clinic or if additional steps are added in the analysis procedure. By serializing and storing a modeled tool sequence, it is even possible to go back to previous versions and to recreate a specific analysis result.

Regarding the temporal coupling of tools in general, the experts were relieved from having to search for each required tool and starting it independently. Although, all data funneling and parameter synchronization still had to be done by hand, the modeled tool sequence and automated tool activation already helped them to focus on the actual data processing and analysis. Especially, moving back and forth between certain analysis tasks and thus frequently switching between heterogeneous tools had previously required considerable manual effort. The basic guidance through the analysis procedure provided by this first layer of tool coupling ensured that no tool or analysis step was left out by mistake, e.g., forgetting data quality checks and applying respective repairs. In addition, it enabled comparability between different analysis runs by activating tools always in line with the experts' analysis procedure.

5 Providing the Right Data at the Right Time

At the second layer of VA tool coupling, we provide the means for VA tools to exchange data, which is probably most directly associated with the joint use of multiple tools. It should foremost make available the analysis results from one tool in a form that can be read-in by the next tool in the chain – and possibly vice-versa, depending on how the tools are used in conjunction. To provide this functionality, we introduce a library for web-based analysis tools to enable data input and output. We illustrate the usefulness of this functionality by showing how it can be used to create and augment VA tools with this functionality, and how this helped toolchaining in an astronomy use case.

5.1 A Library for Data Exchange between VA Tools

Data exchange between multiple VA tools is not as straightforward as it may seem due to the different types of data being generated: while some tools produce numerical outputs, others produce visual outputs, as it lies in the nature of VA. The challenge is that while numerical output can at any time be transformed into visual output, if another tool requires its input in view-form – the other way around is hardly possible. This leads to the situation where if at a later,

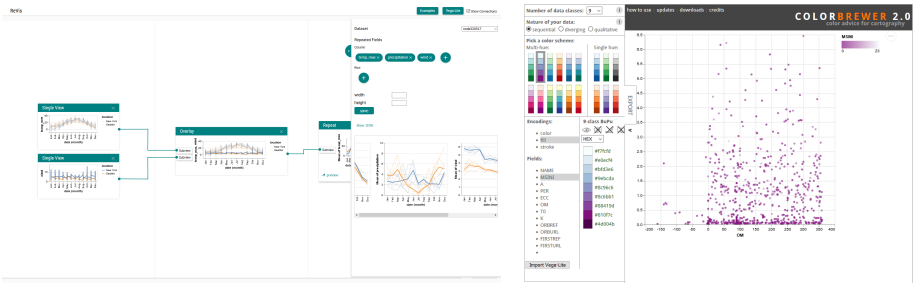


Fig. 4. Screenshots of ReVis (left) and our Vega-Lite enabled ColorBrewer (right).

visual-interactive step of the toolchain it becomes apparent that changes need to be made to analysis steps carried out on the numerical data – e.g., the data should have first been normalized or some aggregation parameter needs to be adjusted – the whole toolchain needs to be rolled back to that point to enact the desired change and then be carried out anew from this point onward.

This challenge is usually countered with a linking between any shown graphical object and its underlying data item – e.g., by employing centralized, MVC-like architectures – so that any interaction and manipulation in the view space can be directly reflected in the underlying data space. This approach works very well, but requires deep integration with the coordinated tools that goes far beyond the mere input and output of files. To instead realize a lightweight data exchange that follows the idea of pairwise connections between tools through passing of files, we propose a different approach: we use the visualization grammar Vega-Lite [33] to describe data, data transformations, visual mapping, and view transformations – i.e., the full visualization pipeline – all at once in one file. Consequently, VA tools that are able to read and write such grammars can adjust any aspect of a visualization, from the underlying dataset all the way to color scales and axis labels in any order.

We support the realization of "Vega-Lite enabled" VA tools by providing the open source library *ReVize* that equips web-based tools with Vega-Lite import/export functionality [15]. ReVize provides an abstraction of Vega-Lite encoded visualizations by making the contained view hierarchy and data transformation graph accessible. To do so, it uses Vega-Lite's view composition structure and decomposes it into independent layers for individual modification. ReVize's import module resolves structural dependencies (e.g., inline datasets or inherited visual encodings) that usually prevent the reuse of sub-views in Vega-Lite specifications by inferring default values from parent and child nodes. A VA tool using ReVize can thus import a Vega-Lite formatted visualization description, further process its contained data or interactively adjust the view on the data it describes, and export it as a Vega-Lite specification again.

ReVize can on one hand be used to build dedicated VA tools that are tailored to processing or adjusting individual aspects captured by the Vega-Lite format. For example, we utilized ReVize for building a data editor called *Re-*

Data to insert/remove data transformations and a visualization editor called *ReVis* to create/adapt the visual mapping and thus the display of the data – cf. Fig. 4(left). On the other hand, ReVize can also be used to add Vega-Lite input/output functionality to pre-existing tools, thus enabling us to use them as part of a Vega-Lite based VA toolchain. We have used ReVize for instance to augment the well-known ColorBrewer web application [13] with Vega-Lite import and export functionality, so that it can be used to adjust color scales in Vega-Lite formatted visualization data. This is shown in Fig. 4(right), and all mentioned tools are furthermore also available for download from https://vis-au.github.io/toolchaining/. Together with all the tools that already feature Vega-Lite input and/or output, these VA tools form the toolset from which analysts can freely pick the most appropriate one to carry out their next analysis step. This is illustrated in the following with an analysis scenario from astronomy.

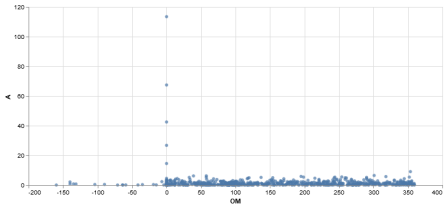
5.2 A Cross-Tool Analysis Scenario for Astronomical Data

In this use case, we worked with a dataset on extrasolar planets publicly available through the Exoplanet Data Explorer at http://exoplanets.org [12]. It contains data on more than 3,000 confirmed planets that were discovered by international research groups before 2018. For each entry, the dataset includes information on the first peer-reviewed publication for a planet, its position as well as physical attributes such as the planet's mass. The dataset is a popular reference in exoplanetary research projects.

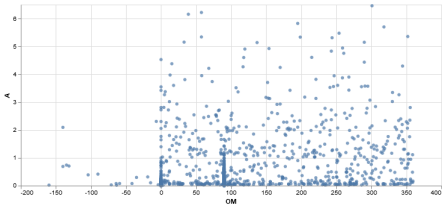
Using this dataset, we want to investigate the planets' masses and their semi major axis, which is a positional parameter that expresses the greatest distance a planet has on its orbit around the star of its solar system. In addition to those, we are also interested in the influence of the so-called argument of periastron, i.e. the angle between the semi major axis and the direction, from which a telescope observed the planet. Looking at these values is helpful in determining characteristics of the types of measurement used to observe these planets.

For investigating this data, we rely on our Vega-Lite based toolset of ReVis, ReData, the Vega-Lite enabled ColorBrewer, and the Vega online editor at https://vega.github.io/editor/. One possible exploration path across these tools is depicted in Fig. 5.

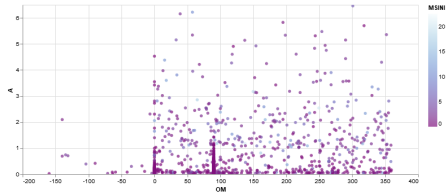
Benefits of coupling tools through data exchange: Besides the advantages that we have already taken into account at design time, such as the expressiveness of Vega-Lite for numerical and image data alike, there are more benefits from data exchange that emerged when working with this use case. First, by using a generic data exchange format that is not tailored to a particular application domain, we were able to bring domain-specific tools like ColorBrewer into the toolchain, even though it is originally intended for use in cartographic applications. Second, by capturing all aspects of an intermediate result from the toolchain in one file, we could simply email these files back and forth with our collaborator from the physics and astronomy department to clarify questions or have him make adjustments to the data part of the file with his specialized tools – mostly IDL



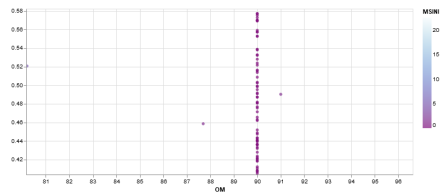
(a) ReVis is used to create a scatterplot of the exoplanets dataset, but outliers at 0° compress the view of the data vertically.



(b) Filtering these outliers using ReData reveals that many of the planets concentrate hardly discernable at 0° and 90°.



(c) Using ColorBrewer to color-code each planet's mass, does not help to better discern individual planets in these spots.



(d) Only zooming into these spots using the Vega online editor finally allows to investigate those planets individually.

Fig. 5. A possible path through a toolset of four independent tools that are coupled by Vega-Lite based data exchange.

scripts. And finally, a simple form of cross-tool undo/redo functionality comes for free with this type of coupling, as snapshots of files can easily be archived every time they are passed into the next tool.

6 Providing the Right Parameters at the Right Time

At the last layer of tool coupling, we define presets for parameters or functions that are used within the analysis toolchain of our application domain. This includes the identification of tasks and their translation into modular execution steps. The Fraunhofer IGD in Rostock has already worked on multiple use case scenarios in the fields of industry, healthcare, and administrative management. One of the most popular commercial projects is the *Plant@Hand* framework [1]. This product is the fundamental structure for multiple descendants such as *Health@Hand* or *Ship@Hand* (see Fig. 6). All of these products are using presets in order to assist users in their daily task. We explain this concept with an example of a healthcare use case in *Health@Hand*.

6.1 A Comprehensive Application for the Visualization of Different Presets

The digitization in the health care sector has led to a growing quantity and quality of individual systems such as the clinical information system (CIS), the

Fig. 6. Examples of visual data analysis use cases with *Plant@Hand*: factory visualization (left) and ship condition analysis (right).

picture archiving and communication system (PACS), the radiological or laboratory information systems (RIS/LIS), or the electronic health record systems (EHR). Additionally, there are specific third-party tools for the analysis and visualization of real-time sensor data as well as customized information dashboards. As a result, the medical staff needs to take on the role of data scientists in order to understand the correlations between multiple tools, observe changes in the data, and assure the well-being of multiple patients at the same time.

Our *Health@Hand* interface assists this cause by providing a 3D model of an intensive care unit (ICU) in order to emphasize the changes of real-time vital data. This digital twin of the hospital ward contains different types of information about patients, rooms, staff, or inventory. We define these categories as *data sources* that each hold a set of *elements* to be handled by the *data manager*. An example of this would be the data source patient that holds elements such as pulse, heart rate variability, or respiration rate.

The visual interface is based on the nested UI model by providing a general overview on the 3D scene with relevant short information about each element. The user can interact with this UI by selecting a single short information panel to open up more detailed information. The detail information panel can contain text, images, or entire tools to further examine selected instances of a data source. The individual tools can be used independently to handle the data from typical healthcare management systems.

- The *Vital Data (VD) Dashboard* shows a patient's vital data parameters such as pulse, heart rate variability, or respiration rate.
- The *eHealth Record Browser* gives the medical staff access to the medical EHR system. This record contains context information about the patient's personal data such as name, age, and prior diagnosis.
- The *Image Viewer* uses the PACS suite to import images of medical findings.
- The *Shimmer Monitor* is used to access readings from an activity sensor. This sensor captures real-time data of the patient's vital parameters and determines the estimated activity.

– The *Cardio Anomaly (CA) Detector* uses artificial intelligence for the analysis of heart rate data. Medical experts can interact with this tool by inspecting and annotating automatically identified anomalous spots in order to point out reasons for a patient's irregular blood circulation.

6.2 Toolchain and Presets for the Detection of Cardio-Vascular Anomalies

Our *Health@Hand* system uses different sorts of parameters in the data manager to control for example 1) which information are shown in the 3D model, 2) how colors are adjusted to avoid visual clutter, 3) how screen space is allocated and organized for the parallel usage of multiple tools, and 4) which tools are linked in order to identify correlations in the visual representation.

The previously mentioned data manager can be used to switch between different data sources. However, there is also a possibility to select or deselect different parameters as every element of a data source is passed as a parameter to create the short information panels. The user can thereby interact with a side menu in order to avoid visual clutter.

Fig. 7. Toolchain view of *Health@Hand* with the toolchain window (bottom right), the history window (bottom left) and the currently used tools (center).

In the following part, we consider the toolchain module that is used for the execution of our specific use case scenario (see Fig. 7). The module itself is a part of the data manager and consists of three different components:

The **Toolchain Window** provides information about the current progress of the user by showing an overview of the analytical toolchain graph. The graph can be explored by interacting with the buttons below in order to switch back and forth in the analytical process. The user can also find useful information about upcoming tools by hovering over the related step.

The **History Window** holds a record of all performed operations whether it is inside of a tool or by interacting with the toolchain window. These operations can then be reverted or repeated on demand.

The **Workspace Area** allows users to interact with the corresponding tools of each toolchain step. It adapts the size of each tool according to the screen space and organizes them next to each other to avoid overlap. The referred data for the scaling and positioning of each tool is thereby passed as a parameter at the initiation of the next toolchain step.

We use this module to represent the analytical process for the detection of cardio-vascular anomalies. The resulting toolchain is based on a specific temporal order of the previously mentioned tools. In the following, we describe our use case in six different steps:

1. **Situational assessment using the three-dimensional *digital twin*:** The medical expert observes and examines the general condition of multiple patients based on their incoming sensor data. If the vital data parameters change to an unusual state, the expert can switch to the toolchain module and select the specific patient. Thereby the patient's ID is recorded and passed through the toolchain in the following process, so that each tool opens up with the right medical data.
2. **Vital data overview using the *VD Dashboard*:** At the first step of the toolchain, a backlog of a patient's vital signs is examined through diagrams and gauges in the *VD Dashboard*.
3. **Evaluation of the personal health state using the *eHealth Record Browser*:** In the next step, the patient record entries such as *age*, *data of admission*, and *diagnosis* are considered in order to put the vital signs in the context of the patient's personal health state. Therefore, the workspace is reduced to show the vital data parameters next to the personal health data.
4. **Checking diagnostic findings using the *Image Viewer*:** The *ImageViewer* provides further context information for the medical expert to check available imaging data for related cardiovascular findings. Therefore, each tool window in the workspace is further reduced to an acceptable size.
5. **Activity analysis using the *Shimmer Monitor*:** As both, pulse and respiration rate, correlate with the physical activity being performed, the patient's activity data (including any therapeutic stressing situations) is brought up from the *Shimmer Monitor*. At this step, Linking & Brushing is established in order to find correlated information based on the selected time interval in the *Shimmer Monitor*. The interval is then passed as a parameter to the *VD Dashboard* and highlighted accordingly.

6. **Automated anomaly detection using *CA-detect*:** If the expert finds no natural cause for the current irregularities, he can use the *cardiological anomaly detection* to calculate anomaly scores based on the vital data streams of the patient. After some minor configuration time, the results are automatically opened in a D3.js line chart to show the statistical outliers.

The use case shows that there are multiple possibilities for using preset parameters in our *Health@Hand* system. We received positive feedback from different visitors at the Medica World Forum for Medicine 2018 and on many following exhibitions. The received feedback included comments such as "That's the future of clinical data exploration." (from *PAMB*) or "The integration of different data views will improve diagnosis and therapy management." (from *Poly-Projekt GmbH*). Especially, changes in the tools itself such as the Linking & Brushing between tools proved to be very helpful for the user. There have also been some minor remarks on the potential of multi-user interaction or the distribution of information across different devices, and we look forward to include these suggestions in further development.

7 Conclusion

With the proposed layered approach for lightweight toolchaining in VA, we have captured in a structured manner the stitching together of VA tools via custom scripts and copy & paste keystrokes that is the current state of affairs in coupling multiple independent VA tools in many domains. The proposed layered model that clearly separates the different aspects of working with ensembles of multiple independent tools, as well as its underlying principle of lightweightedness – i.e., their pairwise, bottom-up coordination – offer a conceptual framework in which to discuss and design toolchains and their desired and necessary degree of automation for a scenario at hand. Finally, the concrete realizations and use cases discussed for each level of coordination show the proposed approach in action, illustrating what each of them has to offer and providing arguments for the usefulness of each of them.

What these different use cases do not yet explicitly discuss, is the potential for the three levels of coordination to play together and to extend into each other's domain. For example, given an encompassing data description like a visualization grammar, adjustments to the control flow can easily be envisioned to reach well beyond mere reparametrizations by switching out not just individual values but entire parts of that description – i.e., parts of the dataset or functional building blocks in the view composition. Or given a planning environment for explicitly creating and revising the usage flow, not only tools and their order of use, but also parameters associated with particular analysis steps can be preconfigured as "temporal presets". In particular such combinations of the different coordination levels will be in the focus of future work on our layered approach.

Acknowledgements

We are indebted to the respective domain experts from ophthalmology, astronomy, and intensive care who provided their feedback and insights for Sections 4 through 6. We also thank the anonymous reviewers for their guiding comments on earlier versions of this manuscript. Furthermore, we gratefully acknowledge funding support by the German Research Foundation through DFG project UniVA.

References

1. Aehnelt, M., Schulz, H.J., Urban, B.: Towards a contextualized visual analysis of heterogeneous manufacturing data. In: Advances in Visual Computing: Proc. of the Intl. Symposium on Visual Computing (ISVC). pp. 76–85. Springer (2013). https://doi.org/10.1007/978-3-642-41939-3_8
2. Benzaken, V., Fekete, J.D., Hémery, P.L., Khemiri, W., Manolescu, I.: EdiFlow: Data-intensive interactive workflows for visual analytics. In: Proc. of the IEEE Intl. Conf. on Data Engineering (ICDE). pp. 780–791 (2011). https://doi.org/10.1109/ICDE.2011.5767914
3. Collins, C., Carpendale, S.: VisLink: revealing relationships amongst visualizations. IEEE TVCG **13**(6), 1192–1199 (2007). https://doi.org/10.1109/TVCG.2007.70521
4. Dörk, M., Carpendale, S., Collins, C., Williamson, C.: VisGets: Coordinated visualizations for web-based information exploration and discovery. IEEE TVCG **14**(6), 1205–1212 (2008). https://doi.org/10.1109/TVCG.2008.175
5. Endert, A., Hossain, M.S., Ramakrishnan, N., North, C., Fiaux, P., Andrews, C.: The human is the loop: new directions for visual analytics. Journal of Intelligent Information Systems **43**(3), 411–435 (2014). https://doi.org/10.1007/s10844-014-0304-9
6. Fekete, J.D.: Visual analytics infrastructures: From data management to exploration. Computer **46**(7), 22–29 (2013). https://doi.org/10.1109/MC.2013.120
7. Fekete, J.D., Hémery, P.L., Baudel, T., Wood, J.: Obvious: A meta-toolkit to encapsulate information visualization toolkits – one toolkit to bind them all. In: Proc. of the IEEE Conf. on Visual Analytics Science and Technology (VAST). pp. 91–100. IEEE (2011). https://doi.org/10.1109/VAST.2011.6102446
8. Fernández-Villamor, J.I., Blasco-García, J., Iglesias, C.A., Garijo, M.: A semantic scraping model for web resources – applying linked data to web page screen scraping. In: Proc. of Intl. Conf. on Agents and Artificial Intelligence (ICAART). pp. 451–456. SciTePress (2011). https://doi.org/10.5220/0003185704510456
9. Fisher, D., Drucker, S., Fernandez, R., Ruble, S.: Visualizations everywhere: A multiplatform infrastructure for linked visualizations. IEEE TVCG **16**(6), 1157–1163 (2010). https://doi.org/10.1109/TVCG.2010.222
10. Garg, S., Nam, J.E., Ramakrishnan, I.V., Mueller, K.: Model-driven visual analytics. In: Proc. of the IEEE Symposium on Visual Analytics Science and Technology (VAST). pp. 19–26. IEEE (2008). https://doi.org/10.1109/VAST.2008.4677352
11. Gürdür, D., Asplund, F., El-khoury, J., Loiret, F.: Visual analytics towards tool interoperabilty: A position paper. In: Proc. of the Intl. Conf. on Information Visualization Theory and Applications (IVAPP). pp. 139–145. SciTePress (2016). https://doi.org/10.5220/0005751401390145

12. Han, E., Wang, S.X., Wright, J.T., Feng, Y.K., Zhao, M., Fakhouri, O., Brown, J.I., Hancock, C.: Exoplanet orbit database. II. updates to exoplanets.org. Publications of the Astronomical Society of the Pacific **126**(943), 827–837 (2014). https://doi.org/10.1086/678447
13. Harrower, M., Brewer, C.A.: ColorBrewer.org: An online tool for selecting colour schemes for maps. The Cartographic Journal **40**(1), 27–37 (2003). https://doi.org/10.1179/000870403235002042
14. Hartmann, B., Doorley, S., Klemmer, S.R.: Hacking, mashing, gluing: Understanding opportunistic design. IEEE Pervasive Computing **7**(3), 46–54 (2008). https://doi.org/10.1109/MPRV.2008.54
15. Hogräfer, M., Schulz, H.J.: ReVize: A library for visualization toolchaining with Vega-Lite. In: Proc. of the Conf. on Smart Tools and Applications in Graphics (STAG). pp. 129–139. Eurographics (2019). https://doi.org/10.2312/stag.20191375
16. Kandel, S., Heer, J., Plaisant, C., Kennedy, J., van Ham, F., Riche, N.H., Weaver, C., Lee, B., Brodbeck, D., Buono, P.: Research directions in data wrangling: Visualizations and transformations for usable and credible data. Information Visualization **10**(4), 271–288 (2011). https://doi.org/10.1177/1473871611415994
17. Kolojejchick, J., Roth, S.F., Lucas, P.: Information appliances and tools in visage. IEEE Computer Graphics and Applications **17**(4), 3–41 (1997). https://doi.org/10.1109/38.595266
18. Krüger, R., Herr, D., Haag, F., Ertl, T.: Inspector Gadget: Integrating data preprocessing and orchestration in the visual analysis loop. In: EuroVis Workshop on Visual Analytics (EuroVA). pp. 7–11. Eurographics Association (2015). https://doi.org/10.2312/eurova.20151096
19. Ludäscher, B., Lin, K., Bowers, S., Jaeger-Frank, E., Brodaric, B., Baru, C.: Managing scientific data: From data integration to scientific workflows. In: Sinha, A.K. (ed.) Geoinformatics: Data to Knowledge, pp. 109–129. GSA (2006). https://doi.org/10.1130/2006.2397(08)
20. Munzner, T.: Visualization Analysis & Design. CRC Press (2014)
21. North, C., Conklin, N., Indukuri, K., Saini, V., Yu, Q.: Fusion: Interactive coordination of diverse data, visualizations, and mining algorithms. In: Extended Abstracts of ACM SIGCHI'03. pp. 626–627. ACM (2003). https://doi.org/10.1145/765891.765897
22. North, C., Shneiderman, B.: Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In: Proc. of the Working Conf. on Advanced Visual Interfaces (AVI). pp. 128–135. ACM (2000). https://doi.org/10.1145/345513.345282
23. Pietschmann, S., Nestler, T., Daniel, F.: Application composition at the presentation layer: Alternatives and open issues. In: Proc. of Intl. Conf. on Information Integration and Web-based Applications & Services (iiWAS). pp. 461–468. ACM (2010). https://doi.org/10.1145/1967486.1967558
24. Poco, J., Mayhua, A., Heer, J.: Extracting and retargeting color mappings from bitmap images of visualizations. IEEE TVCG **24**(1), 637–646 (2017). https://doi.org/10.1109/TVCG.2017.2744320
25. Prakasam, R.K., Röhlig, M., Fischer, D.C., Götze, A., Jünemann, A., Schumann, H., Stachs, O.: Deviation maps for understanding thickness changes of inner retinal layers in children with type 1 diabetes mellitus. Current Eye Research (2019). https://doi.org/10.1080/02713683.2019.1591463
26. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2019)

27. Roberts, J.C.: State of the art: Coordinated & multiple views in exploratory visualization. In: Proc. of the Intl. Conf. on Coordinated and Multiple Views in Exploratory Visualization (CMV). pp. 61–71. IEEE (2007). https://doi.org/10.1109/CMV.2007.20
28. Rogowitz, B.E., Matasci, N.: Metadata Mapper: A web service for mapping data between independent visual analysis components, guided by perceptual rules. In: Proc. of the Conf. on Visualization and Data Analysis (VDA). pp. 78650I–1–13. SPIE (2011). https://doi.org/10.1117/12.881734
29. Röhlig, M., Schmidt, C., Prakasam, R.K., Schumann, H., Stachs, O.: Visual analysis of retinal changes with optical coherence tomography. The Visual Computer **34**(9), 1209–1224 (2018). https://doi.org/10.1007/s00371-018-1486-x
30. Rungta, A., Summa, B., Demir, D., Bremer, P.T., Pascucci, V.: ManyVis: Multiple applications in an integrated visualization environment. IEEE TVCG **19**(12), 2878–2885 (2013). https://doi.org/10.1109/TVCG.2013.174
31. Santos, E., Lins, L., Ahrens, J., Freire, J., Silva, C.: VisMashup: Streamlining the creation of custom visualization applications. IEEE TVCG **15**(6), 1539–1546 (2009). https://doi.org/10.1109/TVCG.2009.195
32. dos Santos, S., Brodlie, K.: Gaining understanding of multivariate and multidimensional data through visualization. Computers & Graphics **8**(3), 311–325 (2004). https://doi.org/10.1016/j.cag.2004.03.013
33. Satyanarayan, A., Moritz, D., Wongsuphasawat, K., Heer, J.: Vega-Lite: A grammar of interactive graphics. IEEE TVCG **23**(1), 341–350 (2017). https://doi.org/10.1109/TVCG.2016.2599030
34. Schulz, H.J., Nocke, T., Heitzler, M., Schumann, H.: A systematic view on data descriptors for the visual analysis of tabular data. Information Visualization **16**(3), 232–256 (2017). https://doi.org/10.1177/1473871616667767
35. Schulz, H.J., Röhlig, M., Nonnemann, L., Aehnelt, M., Diener, H., Urban, B., Schumann, H.: Lightweight coordination of multiple independent visual analytics tools. In: Proc. of the 10th Intl. Conf. on Information Visualization Theory and Applications (IVAPP). pp. 106–117. SciTePress (2019). https://doi.org/10.5220/0007571101060117
36. Seipp, K., Gutiérrez, F., Ochoa, X., Verbert, K.: Towards a visual guide for communicating uncertainty in visual analytics. Journal of Computer Languages **50**, 1–18 (2019). https://doi.org/10.1016/j.jvlc.2018.11.004
37. Streit, M., Schulz, H.J., Lex, A., Schmalstieg, D., Schumann, H.: Model-driven design for the visual analysis of heterogeneous data. IEEE TVCG **18**(6), 998–1010 (2012). https://doi.org/10.1109/TVCG.2011.108
38. Stuerzlinger, W., Chapuis, O., Phillips, D., Roussel, N.: User interface façades: Towards fully adaptable user interfaces. In: Proc. of the ACM Symposium on User Interface Software and Technology (UIST). pp. 309–318. ACM (2006). https://doi.org/10.1145/1166253.1166301
39. Tan, D.S., Meyers, B., Czerwinski, M.: WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In: Extended Abstracts of CHI'04. pp. 1525–1528. ACM (2004). https://doi.org/10.1145/985921.986106
40. Tobiasz, M., Isenberg, P., Carpendale, S.: Lark: Coordinating co-located collaboration with information visualization. IEEE TVCG **15**(6), 1065–1072 (2009). https://doi.org/10.1109/TVCG.2009.162
41. Tominski, C.: Event-Based Visualization for User-Centered Visual Analysis. Ph.D. thesis, University of Rostock, Germany (2006)

42. Waldner, M., Puff, W., Lex, A., Streit, M., Schmalstieg, D.: Visual links across applications. In: Proc. of Graphics Interface (GI). pp. 129–136. Canadian Information Processing Society (2010)
43. Weaver, C.: Visualizing coordination in situ. In: Proc. of IEEE InfoVis'05. pp. 165–172. IEEE (2005). https://doi.org/10.1109/INFVIS.2005.1532143