Lightweight Coordination of Multiple Independent Visual Analytics Tools

Hans-Jörg Schulz[1], Martin Röhlig[2], Lars Nonnemann[3],
Mario Aehnelt[3], Holger Diener[3], Bodo Urban[3], and Heidrun Schumann[2]
[1]*Department of Computer Science, Aarhus University, Åbogade 34, 8200 Aarhus N, Denmark*
[2]*Institute of Computer Science, University of Rostock, Albert-Einstein-Straße 22, 18059 Rostock, Germany*
[3]*Fraunhofer Institute of Computer Graphics Research, Joachim-Jungius-Straße 11, 18059 Rostock, Germany*

hjschulz@cs.au.dk, martin.roehlig@uni-rostock.de, lars.nonnemann@vcric.igd-r.fraunhofer.de,
{mario.aehnelt, holger.diener, bodo.urban}@igd-r.fraunhofer.de, heidrun.schumann@uni-rostock.de

Keywords: Visual Analytics, Software Integration, View Coordination, Unified Interface

Abstract: With the advancement of Visual Analytics (VA) and its spread into various application fields comes along a specialization of methods and tools. This adds complexity and requires extra effort when devising domain-dependent VA solutions, as for every new domain question a new specialized tool or framework must be developed. In this paper, we investigate the possibility of using and re-using existing tools – domain-dependent and general-purpose – by loosely coupling them into specialized VA tool ensembles as needed. We call such coupling among independent tools *lightweight coordination*, as it is minimally-invasive, pair-wise, and opportunistic in utilizing whichever interface a VA tool offers. We propose the use of lightweight coordination for managing the workflow, the data flow, and the control flow among VA tools, and we show how it can be supported with suitable setups of the multiple tool UIs involved. This concept of lightweight coordination is exemplified with a health care scenario, where an ensemble of independent VA tools is used in a concerted way to pursue the visual analysis of a patient's troublesome vital data.

1 INTRODUCTION

The current tool landscape of Visual Analytics (VA) is a scattered one. Most VA tools are specialized, domain-dependent, and scenario-driven solutions, that are tailored to serve their intended analytical purpose in a given application field as best as possible. While this approach caters well to the respective domain experts, it comes at a cost for the VA researcher. Not only does it take a lot of resources and efforts to create, adapt, and fine-tune matching VA tools for every domain problem we encounter. This approach also locks-in algorithms and techniques in highly domain-specific tools, even though they would be useful in other contexts as well.

Looking back at the wealth of VA tools produced by his students over the years, Jarke van Wijk captured these two points as challenges in his VISIGRAPP 2017 keynote[1] by asking "How to develop such kind of custom solutions efficiently?" and "How to scale? How to generalize?" And in fact, these questions are tied to each other: A generalization of domain specific solutions into a generic one can also serve as a formidable basis to jump start the development of a custom solution, which is much more efficient than building it from scratch.

There are two common ways to answer these questions. On one hand, individual VA techniques and functionalities are combined into integrated VA platforms in which to pursue a wide range of visual analyses by utilizing the standard views and computations they incorporate. On the other hand, individual VA tools are simply invoked and used as is in sequence or in parallel. Data and intermediary results are exported from one tool and manually imported into the next via the file system or the clipboard, with multiple tools often being arranged side by side instead of using coordinated views.

In between these two options, there is a third and less traveled road of lightweight coordination of independent VA tools, to form loose multi-tool ensem-

[1] http://www.visigrapp.org/KeynoteSpeakers.aspx?y=2017

bles. These ensembles offer a compromise realizing some of the features of integrated VA platforms with minimal effort on top of independent VA tools. These features can range from lightweight brushing & linking to blending of different VA tools and techniques into one virtual application.

In the spirit of v.Wijk's questions, this paper asks what we can do to provide and facilitate such lightweight coordination between VA tools? To answer this question, this paper introduces, discusses, and exemplifies a novel process model to realize coordination among multiple tools within and across VA systems. This contribution consist of the following four aspects:

- a 3-stage approach for defining lightweight coordination among multiple VA tools by means of augmenting existing analysis workflows in Sec. 3;
- a decentralized model based on this process that captures coordination in VA tool ensembles through pairwise linkage of tools in Sec. 4;
- a collection of prototypical UI setups that permit a central, unified access to such a loosely coupled, decentralized tool ensemble in Sec. 5;
- an application example from health analytics showcasing the proposed design process, coordination model, and UI setups in Sec. 6.

2 RELATED WORK

Prior research on tool coordination has resulted in a variety of frameworks such as *OBVIOUS* (Fekete et al., 2011) or *VisMashup* (Santos et al., 2009), as well as in application-specific interoperability standards like *BioJS* (Gómez et al., 2013) for the life sciences or *SAMP* (Taylor et al., 2015) for astronomy. They provide an interoperability layer on code level, which offers the necessary functionality to software developers for coupling their VA tool with other tools using the same framework. Lightweight coordination among tools tries to keep code changes to a minimum and instead relies on tool synchronization on *data level* and on tool integration on *view level*.

Data level coordination in visualization – i. e., the synchronization of visualization tools through data exchange – dates back to 1997 (cf. *Visage* (Kolojejchick et al., 1997)). The traditional approach to data level coordination is based on a central database, that acts as a model in the model-view-controller mechanism. Examples are systems like *Snap-together* (North and Shneiderman, 2000) and *EdiFlow* (Benzaken et al., 2011), that both use an underlying relational database as central storage for the information exchange. Other applications such as the *Metadata Mapper* (Rogowitz and Matasci, 2011) use services and communication protocols to share information among different tools. Besides these often centralized systems, there are also decentralized systems such as federated databases or mashup tools, such as *Mashroom+* (Liu et al., 2014). Some tools even use custom connectors to pass data from and to otherwise closed tools with mere *screen poking* and *screen scraping* (Fernández-Villamor et al., 2011; Hartmann et al., 2008) mechanisms.

View level coordination can be achieved by exchanging and combining user interface (UI) components and visualizations from different systems, as it is proposed by approaches, such as *ManyVis* (Rungta et al., 2013). As the coordination of applications on data and view level is independent, they can be combined as necessary to achieve the required flexibility for a given analysis. One of the few examples for coordination on both levels is *Webcharts* (Fisher et al., 2010). To make the interoperability of the tools available to the user, standard visualization frameworks usually display such an interlinked setup in multiple coordinated views (Dörk et al., 2008; Roberts, 2007) or fused visualizations (North et al., 2003). In some cases, the tool chain itself is visually encoded in a (directed) graph layout that shows their connections (Tobiasz et al., 2009; Gürdür et al., 2016). Yet in some cases, arranging UIs side-by-side is not sufficient and a common interface may be glued together from individual interface parts. *WinCuts* (Tan et al., 2004) and *Façades* (Stuerzlinger et al., 2006) enable users to interactively compose their own UI by cutting out pieces of existing applications and re-arranging them. For web-based ensembles, existing UIs are combined in mashups (Pietschmann et al., 2010).

In most application domains from climatology to biomedicine, the current practice in data analysis is the "tool chain" of independent VA tools simply being used independently 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 for our coordination approach, which is described in the following.

3 BASIC APPROACH

Our basic idea is to couple multiple VA tools according to existing analysis workflows. To this end, we introduce a 3-stage coordination procedure.

The first stage takes the temporal order of VA tools into account, the second stage enables data exchange, and the third stage aims at synchronizing the VA tools. Defining such a tool coordination requires the combined efforts of a VA expert and a domain expert with knowledge about the workflow in question. Together, they do not only decide where to coordinate a tool chain – i.e., among which tools – but also to which extent to coordinate them.

Coordinating the workflow. The workflow determines which VA tools are used in which combination – i.e., their subsequent or concurrent operation. The first step for a VA expert is to understand this workflow with the help of an application expert in order to identify tool dependencies that should be automated via tool coordination. This step deals with the *situational level* in the nested visualization design model (Munzner, 2014, ch.4.3) that aims to understand the domain situation. Together, VA expert and application expert determine which parts of the tool chain shall be coordinated in the sense of *automatically bringing tools up as they are needed* according to the workflow.

Coordinating the Data Flow. The data flow determines what the VA tools are working on and what they produce – i.e., their inputs and outputs. Where the workflow is determined by the chain of tools used during an analysis, the data flow is determined by the data funneled through this tool chain from the "raw" data input to the first tool of the chain, all the way to the refined result output by the final tool in the chain. This second step can thus be seen as a special case of the *abstraction level* in the nested model that aims to understand the data and what is done with it in a domain-independent way. At this step, VA expert and application expert need to determine for which parts of the tool chain can and should *data be transmitted as the analysis switches between VA tools*.

Coordinating the Control Flow. The control flow determines how the tools are used – i.e., the steering of their inner workings. Where the data flow is determined by the inputs and outputs of VA tools, the control flow is determined by which interactions are performed on them, which methods are selected and which parameters are tuned. To understand these interactive practices, the VA and application experts might need to bring in actual domain users to observe exactly how the different VA tools are used. This third step relates to the *encoding and interaction level* of the nested visualization design model aiming to find out how the general workflow is specifically carried out among tools, beyond the mere passing of data. At this step, VA and application experts must specify if any of these *interactively invoked actions in one tool shall be synchronized in another tool* – and if so, how this synchronization should look like.

The last level of the nested visualization design model, the *algorithmic level*, strongly depends on the concrete VA tools being used. That is why this level is usually the responsibility of a software engineer who implements and thus realizes the concrete mode of coordination to be used. To do so, any suitable technical realization can be employed – from custom application scripting to the utilization of coordination frameworks, such as OBVIOUS or ManyVis.

With our 3-stage approach for tool coordination, we contribute a procedure to model the top three levels of the nested framework. Such a coordination allows to provide a centralized, workflow driven access to VA tools via a unified interface. The details of our model are provided in the following Section 4 and the unified access is described in Section 5.

4 A PROCEDURE FOR MODELING LIGHTWEIGHT COORDINATION

To characterize coordination among VA tools on the three levels of workflow, 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., (Weaver, 2005) and (Collins and Carpendale, 2007), we propose to model VA tool coordination as a graph. In this graph, VA tools constitute the nodes and directed edges capture the workflow, 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.

4.1 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 tool chain. 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. Fig. 1 shows two tools modeled in this way, capturing the following I/O possibilities as *ports* of a VA tool:

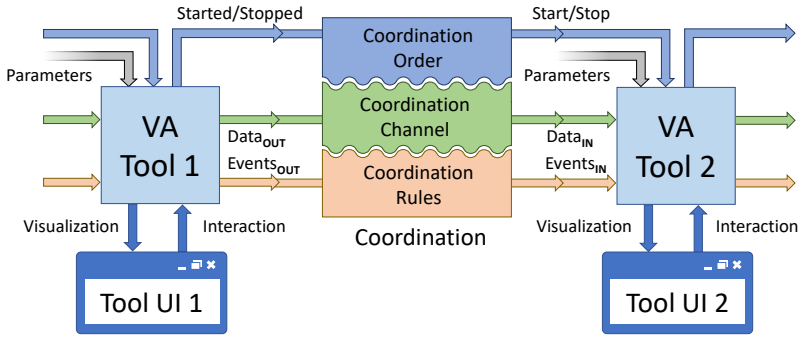

Figure 1: Conceptual abstraction of lightweight coordination between two VA tools. The *coordination order* models any temporal dependency between two tools (i.e., their subsequent or concurrent use). The *coordination channels* capture ways to exchange data between tools, including any necessary data transformations along the way. The *coordination rules* describe automated syncing features between tools as action/reaction pairs.

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*.

Parameters: 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 *Parameters*.

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 $Data_{OUT}$.

Events (in/out): VA tools may allow a certain degree of steering and observing its inner workings. Commands may be passed to a VA tool through the port $Events_{IN}$ – e.g., to run a certain computation – and the progress and success of carrying them out may be passed to the outside world via the port $Events_{OUT}$.

Visualization & Interaction: In most cases, VA tools generate visualizations of the input data and display them to the user, who can then make interactive adjustments to them. These aspects are captured by the ports *Visualization* and *Interaction*.

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 progress and success of an analytic computation (i.e., $Events_{OUT}$) from the updated results (i.e., $Data_{OUT}$). In other cases, we might need the user to step in – for example, by explicitly signaling the completion of a task by closing the corresponding tool.

4.2 Modeling the Pairwise Flow

Between VA tools, coordination is possible on any of the three levels of workflow, data flow, and control flow. This leads to three different edge types that can connect two VA tools – i.e., nodes of the model:

- the *coordination order* describing any automation of the workflow between two VA tools – i.e., showing them subsequently or concurrently;
- a *coordination channel* describing any transmission of the data flow between two VA tools – i.e., passing data between them, including any necessary data transformation along the way;
- and a *coordination rule* describing the synchronization of the control flow between two VA tools – i.e., an (inter)action in one tool generating a reaction in the other tool.

These are schematically depicted in Fig. 1. In the following, we describe each of the three edge types and the aspects of tool coordination they facilitate.

4.2.1 Modeling the workflow

At the lowest level of coordination, there is the temporal order of VA tools – i.e., the workflow determined by the analysis scenario and its domain. The workflow captures the tool chain, that is the succession of

VA tools as they are used by the user in pursuance of a particular analysis goal. It can include subsequently used tools, as well as concurrently used tools. At this level, VA tool coordination brings up the right tools at the right time, as it is foreseen by the workflow.

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

Coordination on this level already provides as much as an automated guidance of the user along the predefined workflow, in the spirit of approaches such as *Stack'n'Flip* (Streit et al., 2012). While the users still have to do everything else themselves – such as moving data back and forth between the tools, or adjusting their visualizations to match up – the coordination order between tools allows to automatically move the VA tool chain 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 workflow. This is important in cases where carrying out analysis steps in different orders yields different results – e.g., when performing a combination of dimension reduction and clustering (Wenskovitch et al., 2018). 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.

4.2.2 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 analyze. 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 tool chain – i.e., the flow of data tends to follow the workflow.

In our model, this coordination is achieved via *coordination channels* – shown in the middle of Fig. 1. These capture the possibility to pass data from a VA tool's output to another's input, as well as performing 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 any data flow-oriented visualization framework does. While the user still has to manage what is to happen with that data in a currently used VA tool – i.e., starting computations and generating 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 and competing standards. 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 cache 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 from the corresponding channels, thus providing a coherent analysis experience forward *and* backward along the tool chain.

Note that "passing data" can be a complex problem and research on data integration and scientific workflows has established various approaches to do so (Ludäscher et al., 2006). For modeling the coordination on data level, it is enough to know that one of these approaches underlies each channel to realize the necessary data transport and transformation.

4.2.3 Modeling the Control Flow

The last level of coordination is about having changes in one VA tool also reflected in other tools. This is probably the most common understanding of coordination, where for example, selecting data items in one tool will also highlight them in other tools. The control flow captures such use of VA tools by observing events that occur in one tool and then triggering corresponding events in another tool. 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 the desired synchronization between two tools as an action/reaction pair that waits for certain events (action) in one tool, and triggers corresponding events (reaction) in another tool:

$$(Tool_1.Events_{OUT} = \texttt{selected}) \Longrightarrow (Tool_2.Events_{IN} = \texttt{highlight})$$

Coordination rules can either rely on coordination channels to transmit any datasets or results that go along with the observed or triggered event, or they can

use direct parameter changes on the VA tools. Coordination on this functional level can be used to provide any of the common coordination patterns, such as linking & brushing, navigational slaving, or visual linking (Waldner et al., 2010). 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 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 to the next tool in the tool chain 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 tool chain.

4.3 Combining Tools and Flows

Taken together, we yield a 3-tiered model of lightweight coordination among VA tools. The model inherently captures the lightweightedness of our coordination approach, as any coordination is added on top of the VA tools without making changes to them, but by utilizing the inputs and outputs they offer.

The proposed model is thus comprised of the VA tools, as well as of the three coordination levels building on top of each other to realize different degrees of coupling among those tools. It can thus be expressed as a 4-tuple $CM = (Tools, WorkFlow, DataFlow, ControlFlow)$ consisting of the following sets:

- The set of *Tools* as it is given by the analysis setup.
- The *WorkFlow*, 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 $(Source, Events_{OUT}, Target, Events_{IN})$ capturing an event in a source tool that triggers another event in a target tool.

All three sets of *WorkFlow*, *DataFlow*, and *ControlFlow* model the pairwise coordination between VA tools: The *WorkFlow* contains the answer to the question of which parts of the tool chain to automate tool-wise and in which order. The *DataFlow* comprises the answer to the question along which parts of the tool chain to transmit data using which data transformation. And the *ControlFlow* captures the answer to the question of which actions (in particular interactions) to pass as events along the tool chain to synchronize tools through reactions.

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 even 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. Note that this approach is very different from the common centralized approaches. Instead of trying to lift a whole tool chain up to conform to a state-of-the-art coordination framework, we seek to 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 domain experts, who have a lower hurdle to adopt coordination as partial automation of their tried-and-true tool chains,
- 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 users, who can bring in additional tools, which may or may not already have couplings to the tools from a current tool chain.

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

5 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 a structured and organized view on the otherwise often overwhelming network of any number of invisible pairwise coordination mechanisms springing to life depending on the currently used tools and the user's actions.

5.1 Individual VA Tool Use: The Wizard or Tabbed UI

The simplest graph topology induced by the tool chain 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 as shown in Fig. 2. What reads like an oversimplification at first is actually the most prevalent usage pattern in practice. From raw data to insight and from overview to detail – 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 wizard interface that leads the user step by step along the path defined by the coordination model. Another variant is a tabbed interface that opens each tool in a dedicated tab, with the tabs being ordered according to the tool sequence. Whether a wizard or a tabbed interface is used, the user is always able to go back in the tool chain 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 tool chain 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.

Figure 2: The path topology of sequential tool use with one tool being shown at a time.

5.2 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 tool chain at once. This can be the case, for example, when a data 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 would still be a path topology, as shown in Fig. 3, but this time 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 as necessary and without having to switch – i.e., sending one to the background and bringing another one to the front, as it would be the case for the tabbed interface. Synchronization features, such as linking & brushing and displaying visual links are desirable to make the back and forth between the two tools even more fluent. These can be captured as coordination rules.

Figure 3: The path topology of sequential tool use with two tools (UI i and UI i+1) being shown simultaneously.

5.3 Flexible VA Tool Use: The UI Mash-up

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. 4 where all analysis steps start from a hub application or central VA tool. Such topologies support more complex workflows 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 embedded 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 opened 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. This map-like appearance can be provided, for example, by a spatial visualization (Butkiewicz et al., 2008).

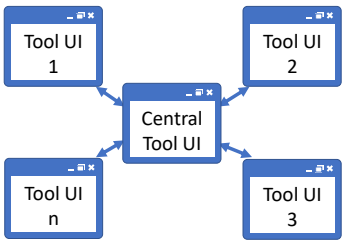

Figure 4: The star topology of multiple tools being used in combination with a central tool UI as mediator.

5.4 The Control Interface

VA tool use in practice can actually combine all of the above patterns, just as they may be needed during a particular stage of an analysis. As illustrated in Fig. 5, it is not uncommon that the same tool may be instantiated multiple times for different parts of the data, or that starting a tool will lead to findings that trigger a whole new sequential analysis workflow. For these cases, there is no principal way of how to best combine all the UIs involved that would be applicable to all possible such topologies, and it has to be negotiated with the domain expert. The more complex this ensemble gets, the more important it becomes to maintain an overview of the analysis and to be able to parametrize and steer it. This is where the *control interface* comes into play.

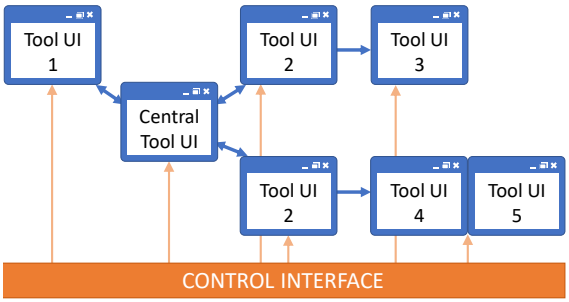

Figure 5: The control interface that serves as a global parametrization facility and for displaying process information. In this context VA tools can be utilized as a combination of single (UI 1 and UI 3), pair-wise (UI 4 and UI 5), and multiple (UI 2) references from a central hub application (Central UI).

A control interface can offer a global display of the analysis workflow and the user's progress in pursuing it (Streit et al., 2012). Yet it can also be used for global parameter settings affecting all opened tools without having to adjust each of them individually. In the following, we give an example of how such an interface ensemble can look like.

6 DEMONSTRATING EXAMPLE

In this section, we demonstrate the conceptual ideas presented in the previous sections by instantiating a tool ensemble – called Health@Hand – for analyzing health data.

6.1 The Analysis Scenario

In today's health care domain, the ongoing digitalization leads to a growing quantity and quality of individual health data. This results in a situation where medical staff turn more and more into data analysts. It is not uncommon anymore that entire hospital wards and in particular the intensive care units (ICUs) are centrally monitored by a head nurse or doctor, who is thus able to keep tabs on the well-being of multiple patients at the same time.

In the scenario we use to illustrate our concept of lightweight coordination, this monitoring is done on a 55 inch multi-touch table in a single-user, non-collaborative setting. A medical professional observes and examines the state of multiple patients as mediated by their incoming vital data. This data arrives in real-time and is guaranteed to be updated in intervals of ≤ 3 seconds. It includes heart rate, heart rate variability, blood pressure, breathing rate, and in some cases also the blood sugar level. In case a patient's vital signs show irregularities, the medical staff needs to cross-check information on that patient to identify possible causes and call-in the appropriate specialist if necessary. So, the analysis task we want to support is two-fold: (1) check for irregularities and (2) investigate possible causes for them.

6.2 Specifying the Tool Set

To pursue the above tasks, medical professionals have a number of VA tools at their disposal. In clinical practice, these are usually part of the clinical information management system (CIS), of the picture archiving and communication system (PACS), of the radiological or laboratory information systems (RIS/LIS), or of the electronic health record systems (EHR). On top of those come individual tools provided by sensor

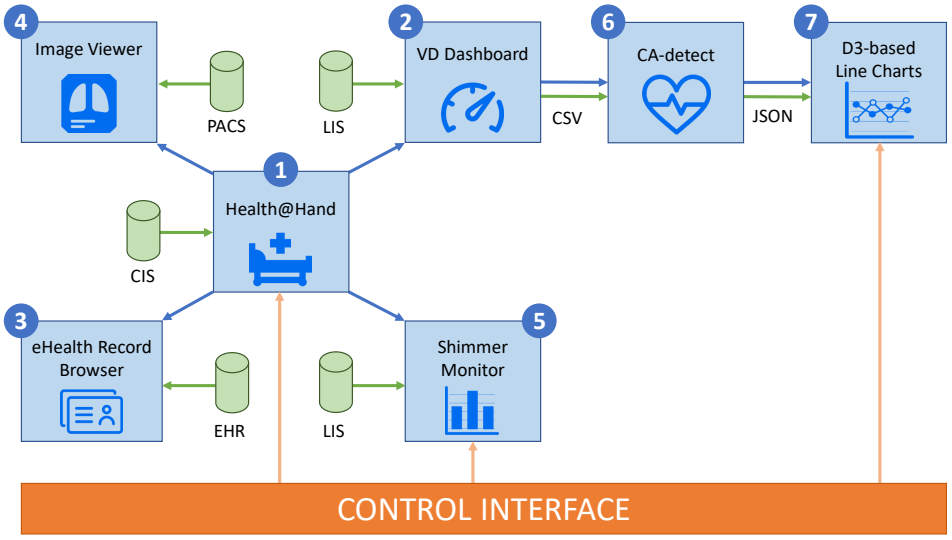

Figure 6: Combined view of a coordination model and the respective UI setup for a medical analysis scenario to detect cardiovascular symptoms and anomalies within a patient's past and present clinical data utilizing multiple interlinked VA tools. The workflow is denoted with blue arrows, the data flow with green arrows, and the control flow with orange arrows.

manufacturers, 3rd party analysis and visualization tools, as well as customized information dashboards for monitoring scenarios specific to a particular hospital ward. The concrete workflow we describe here includes the tools:

- *eHealth Record Browser* to access the EHR,
- *Image Viewer* to access the PACS suite,
- *Shimmer Monitor* to access readings from an activity sensor,
- *VD Dashboard* that shows a patient's vital data,
- *CA-detect* for analyzing heart rate data,
- a custom *D3-based Line Charting* tool to visualize the output from *CA-detect*, and
- the *Health@Hand framework* for visual monitoring of staff and patients.

6.3 Specifying the workflow

In the first stage, the temporal order of using the tools must be determined. As shown in Fig. 6 their application stretches from the monitoring of the entire ward to the specific analysis of a particular patient. We identified seven reoccurring steps – with steps 1 through 5 being dedicated to checking for irregularities, and steps 6 and 7 to investigating possible causes:

1. **Situational assessment using Health@Hand:** The medical expert monitors the general condition and state of assigned patients. From this overview display, the expert can select patients with critical vital data for further examination.
2. **Vital data overview using *VD Dashboard*:** A backlog of a patient's vital signs can be examined through diagrams and gauges in this dashboard. For more context, a patients diagnosis and treatment plan can be opened-up from the *Health@Hand* overview.
3. **Diagnosis details using *eHealth Record Browser*:** From the patient records, the entries *age* and *diagnosis* help the medical expert to put the vital signs in context. An additional display of the outcome of any procedures can be triggered from the *Health@Hand* overview as well.
4. **Checking imaging data using the *Image Viewer*:** The medical expert checks available imaging data relating to the cardiovascular system, as it is the heart rate showing irregularities. As both, pulse and respiration correlate with the physical activity being performed, the patient's activity data can be brought up from the *Health@Hand* overview.
5. **Activity analysis using the *Shimmer Monitor*:** Looking at the patient's recent activities including any therapeutic stressing situations, the expert finds no natural cause for the current irregularities. He thus switches back to the *VD Dashboard* to run an automated anomaly detection.
6. **Automated anomaly detection using *CA-detect*:** The cardiac anomaly detection calculates anomaly scores on the vital data streams. After some minor configuration, the results are automatically opened in a line chart.

7. **Anomaly analysis using the *D3-based Line charting* tool:** The medical expert inspects the identified anomalous spots and annotates sections that point towards reasons for the patient's apparent irregular blood circulation.

6.4 Specifying the Data Flow

In the second stage, we must specify the coordination of the data among the steps of the workflow. The data to be passed between tools consist of two parts: the general medical background information about a patient and the situation-specific data that is currently observed and requires analysis. While the medical background information is available from a number of centralized systems within the hospital and can be queried via the patient's identifier (ID), the situation-specific data is only available locally.

As a result, data is passed between tools in two ways: On one hand for accessing medical background information, the patient's ID is passed as a parameter into the tools, which then query the relevant data records themselves. This variant of a centralized data access is employed by tools such as the *eHealth Record Browser*, the *Image Viewer*, and the *VD Dashboard*. The situation-specific data on the other hand is passed directly from tool to tool, requiring a full-fledged data channel between them. The variant of a decentralized data access is employed by tools such as *CA-detect* and the *D3-based Line Charting*. The passing of this data is denoted in Fig. 6 by a second arrow in green connecting the respective tools.

6.5 Specifying the Control Flow

In the third stage, we specify how the tools are used in the given workflow. In our example, tools are mainly applied subsequently as none of the workflow steps described in Sec. 6.3 requires using multiple tools at once. Yet that does not mean that multiple tools cannot be (left) open on the large touch screen for further reference (e.g., the *Shimmer Monitor* providing the overview of a patient's activities) or easier adjustment of input data (e.g., the *VD Dashboard* for selecting different time intervals of interest to be processed by *CA-detect* and its results being updated in the *D3-based Line Charting* tool.

As independent as these analysis steps and each respective tool comes, they nevertheless share parameters whose adjustment is worthwhile to coordinate globally through a control interface. An example of such a parameter occurs, when the medical expert switches between hourly, daily, and weekly temporal resolutions to find prior incidences on a larger time scale or if the medical expert looks into details of a found incident on a smaller time scale. To specify the control flow, we define a set of appropriately configured rules that synchronize these adjustments between all time-oriented data displays. The temporal resolution can also be adapted globally using a menubar at the bottom of *Health@Hand* as a control interface for the entire tool ensemble.

6.6 Designing the UI for the Modeled workflow

The UI design for the described scenario mirrors the workflow in most aspects. This is why we have integrated both, the coordination model and the UI setup, in a single schematic depiction shown in Fig. 6. Its overall UI topology we apply is akin to the star topology discussed in Sec. 5.4 with *Health@Hand* in the role of the central VA tool. This choice was made to tie-together the UIs of the individual VA tools on top of an inherently spatial "digital twin" of the medical facility to provide sense of locality for the UIs, linking them to the respective patient in question. This can be seen in the screenshot in Fig. 7, where *Health@Hand* is shown in the background of the UI ensemble and other tools are opened on top of it.

The control interface was designed to take the form of an unobtrusive menubar at the bottom of the screen that is always visible and gives access to functionality for managing and parametrizing the UIs. The positioning and appearance of the control interface furthers the impression of the UI ensemble as a desktop environment where tools are centrally managed through a task- or statusbar. By mimicking the desktop metaphor, users feel comfortable and knowledgeable about managing the ensemble without much training. On top of the familiar visual appearance, the functionality of this environment is carefully adjusted to support coordination along the workflow. For example, workflow sensitive taps open up the next tool in the workflow with every tap on the same patient – i.e., the first tap opens the *VD Dashboard*, another tap opens the *eHealth Record Browser* and so on. An interactive walk-through of our UI setup is given in the video that accompanies this paper[2].

6.7 Feedback on the Use of *Health@Hand*

Health@Hand and its lightweight coordination capabilities form a commercially available platform that has been presented to the public at the Medica World

[2]Video DOI: 10.6084/m9.figshare.7571030.v1

Figure 7: Screenshot of *Health@Hand* with the *Shimmer Monitor* (left) and *VD Dashboard* (right) opened on top of it.

Forum for Medicine 2018. User feedback so far has been very encouraging, including comments such as "That's the future of clinical data exploration." (from *PAMB*) or "The integration of different data views from different tools will improve diagnosis and therapy management" (from *Poly-Projekt GmbH*). In addition, users and potential customers also pointed out aspects leaving room for future improvement, such as multi-user workflows and tool UI templates.

7 CONCLUDING REMARKS

The use case shows that with our approach of defining and designing lightweight coordination, we have found one possible answer to the question of *How to efficiently design custom solutions?* – given all experts involved are sharing their knowledge freely or can be persuaded to do so (Vosough et al., 2017). The layered, pairwise structure of our approach allows us to incrementally realize new tool ensembles and to selectively adjust coordination aspects when customizing an existing tool ensemble. At the same time, already centralized aspects of a software landscape (like the patient records in the example) can still be leveraged and others (like the temporal resolution) can be given a central look & feel through the UI as desired. This results in a mixed approach that utilizes the different coordination channels to achieve a least-effort integration among tools, data, and UI. This form of coordination can incorporate any already centralized data coordination and add centralized UI elements by means of the control flow to provide a unified access to this strung-together tool ensemble underneath.

While this approach was a perfect fit for the medical workflow described in the example, it may not be the most suitable coordination approach for all scenarios. In principle, a decentralized coordination can run into all the problems of distributed systems ranging from deadlocks to race conditions. This again underlines the importance of a well-defined workflow to begin with, as that workflow constraints the multitude of possible tool combinations to only those that are actually necessary, thus preventing those issues. So, in terms of the question of *How to generalize?* lightweight coordination, we will have to investigate means to make these workflows more flexible and allow more deviations from them.

ACKNOWLEDGEMENTS

We thank Dieter Schmalstieg and Marc Streit for their input on early versions of the principal idea of lightweight coordination. We are also indebted to Marian Haescher for his work on the accompanying video. This research was supported by the German Research Foundation (DFG). The icons in Figure 6 were made by user Freepik and are used under the

Basic License from www.flaticon.com.

REFERENCES

Benzaken, V., Fekete, J., Hémery, P., Khemiri, W., and Manolescu, I. (2011). EdiFlow: Data-intensive interactive workflows for visual analytics. In *Proc. of ICDE'11*, pages 780–791.

Butkiewicz, T., Dou, W., Wartell, Z., Ribarsky, W., and Chang, R. (2008). Multi-focused geospatial analysis using probes. *IEEE TVCG*, 14(6):1165–1172.

Collins, C. and Carpendale, S. (2007). VisLink: revealing relationships amongst visualizations. *IEEE TVCG*, 13(6):1192–1199.

Dörk, M., Carpendale, S., Collins, C., and Williamson, C. (2008). VisGets: Coordinated visualizations for web-based information exploration and discovery. *IEEE TVCG*, 14(6):1205–1212.

Fekete, J. D., Hémery, P. L., Baudel, T., and Wood, J. (2011). Obvious: A meta-toolkit to encapsulate information visualization toolkits – one toolkit to bind them all. In *Proc. of IEEE VAST'11*, pages 91–100. IEEE.

Fernández-Villamor, J. I., Blasco-García, J., Iglesias, C. A., and Garijo, M. (2011). A semantic scraping model for web resources – applying linked data to web page screen scraping. In *Proc. of ICAART'11*, pages 451–456. SciTePress.

Fisher, D., Drucker, S., Fernandez, R., and Ruble, S. (2010). Visualizations everywhere: A multiplatform infrastructure for linked visualizations. *IEEE TVCG*, 16(6):1157–1163.

Gómez, J., García, L. J., Salazar, G. A., Villaveces, J., Gore, S., García, A., Martín, M. J., Launay, G., Alcántara, R., del Toro, N., Dumousseau, M., Orchard, S., Velankar, S., Hermjakob, H., Zong, C., Ping, P., Corpas, M., and Jiménez, R. C. (2013). BioJS: an open source JavaScript framework for biological data visualization. *Bioinformatics*, 29(8):1103–1104.

Gürdür, D., Asplund, F., El-khoury, J., and Loiret, F. (2016). Visual analytics towards tool interoperabilty: A position paper. In *Proc. of IVAPP'16*, pages 141–147. SciTePress.

Hartmann, B., Doorley, S., and Klemmer, S. R. (2008). Hacking, mashing, gluing: Understanding opportunistic design. *IEEE Pervasive Computing*, 7(3):46–54.

Kolojejchick, J., Roth, S. F., and Lucas, P. (1997). Information appliances and tools in visage. *IEEE Computer Graphics & Applications*, 17(4):3–41.

Liu, C., Wang, J., and Han, Y. (2014). Mashroom+: An interactive data mashup approach with uncertainty handling. *Journal of Grid Computing*, 12(2):221–244.

Ludäscher, B., Lin, K., Bowers, S., Jaeger-Frank, E., Brodaric, B., and Baru, C. (2006). Managing scientific data: From data integration to scientific workflows. In Sinha, A. K., editor, *Geoinformatics: Data to Knowledge*, pages 109–129. GSA.

Munzner, T. (2014). *Visualization Analysis & Design*. CRC Press.

North, C., Conklin, N., Indukuri, K., Saini, V., and Yu, Q. (2003). Fusion: Interactive coordination of diverse data, visualizations, and mining algorithms. In *Ext. Abstracts of ACM SIGCHI'03*, pages 626–627. ACM.

North, C. and Shneiderman, B. (2000). Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proc. of AVI'00*, pages 128–135. ACM.

Pietschmann, S., Nestler, T., and Daniel, F. (2010). Application composition at the presentation layer: Alternatives and open issues. In *Proc. of iiWAS'10*, pages 461–468. ACM.

Roberts, J. C. (2007). State of the art: Coordinated & multiple views in exploratory visualization. In *Proc. of CMV'07*, pages 61–71. IEEE.

Rogowitz, B. E. and Matasci, N. (2011). Metadata Mapper: A web service for mapping data between independent visual analysis components, guided by perceptual rules. In *Proc. of VDA'11*, pages 78650I–1–13. SPIE.

Rungta, A., Summa, B., Demir, D., Bremer, P. T., and Pascucci, V. (2013). ManyVis: Multiple applications in an integrated visualization environment. *IEEE TVCG*, 19(12):2878–2885.

Santos, E., Lins, L., Ahrens, J., Freire, J., and Silva, C. (2009). VisMashup: Streamlining the creation of custom visualization applications. *IEEE TVCG*, 15(6):1539–1546.

Streit, M., Schulz, H.-J., Lex, A., Schmalstieg, D., and Schumann, H. (2012). Model-driven design for the visual analysis of heterogeneous data. *IEEE TVCG*, 18(6):998–1010.

Stuerzlinger, W., Chapuis, O., Phillips, D., and Roussel, N. (2006). User interface façades: Towards fully adaptable user interfaces. In *Proc. of ACM UIST'16*, pages 309–318. ACM.

Tan, D. S., Meyers, B., and Czerwinski, M. (2004). WinCuts: Manipulating arbitrary window regions for more effective use of screen space. In *Ext. Abstracts of ACM SIGCHI'04*, pages 1525–1528. ACM.

Taylor, M. B., Boch, T., and Taylor, J. (2015). SAMP, the Simple Application Messaging Protocol: Letting applications talk to each other. *Astronomy and Computing*, 11(B):81–90.

Tobiasz, M., Isenberg, P., and Carpendale, S. (2009). Lark: Coordinating co-located collaboration with information visualization. *IEEE TVCG*, 15(6):1065–1072.

Vosough, Z., Groh, R., and Schulz, H.-J. (2017). On establishing visualization requirements: A case study in product costing. In *Short Paper Proc. of EuroVis'17*, pages 97–101. Eurographics Association.

Waldner, M., Puff, W., Lex, A., Streit, M., and Schmalstieg, D. (2010). Visual links across applications. In *Proc. of GI'10*, pages 129–136. Canadian Information Processing Society.

Weaver, C. (2005). Visualizing coordination in situ. In *Proc. of IEEE InfoVis'05*, pages 165–172. IEEE.

Wenskovitch, J., Crandell, I., Ramakrishnan, N., House, L., Leman, S., and North, C. (2018). Towards a systematic combination of dimension reduction and clustering in visual analytics. *IEEE TVCG*, 24(1):131–141.