Thomas Geymayer

Exploring the information worker's space

Doctoral Thesis

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisors

Prof. Dr. Dieter Schmalstieg
Institute for Computer Graphics and Vision, Graz University of Technology

Prof. Dr. Harald Reiterer
Human-Computer Interaction Group, University of Konstanz

Graz, July 2019

TO MARTA

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Acknowledgments

This thesis was only possible due to the support of several people. First of all, I want to thank my supervisor Prof. Dieter Schmalstieg, who enabled me to do this thesis in the first place, and gave me the freedom to follow my research interests, as well as providing valuable feedback and support when I needed it. I also want to thank Prof. Harald Reiterer for being my second supervisor and providing constructive feedback for my thesis.

I must further thank Alexander Lex and Marc Streit, who especially during their time in Graz closely supported me, but also later continued to provide feedback and support my papers. Likewise, I am grateful to Markus Steinberger who supported me in setting up the project, and, especially for his support in GPU programming. Special thanks go to Manuela Waldner, for supporting the design and evaluation of the user studies.

Besides the creators behind the many enabling technologies and tools I want to explicitely mention Werner Puff and Mark Doktor, whose implementations were very useful to start my development. Also Daniel Brajko was very supportive in setting up and maintaining different hardware. Furthermore, I have to thank all the participants of the user studies for providing their time and valuable feedback.

I am very grateful to my family, that enabled me to start my studies and supported my while following my interests. Finally, I want to dedicate this thesis to Marta, who supported me to finish this thesis and at the time of the final presentation will be my wife.

Abstract

Extracting facts out of document collections (*information foraging*) and synthesizing knowledge (*sensemaking*) are tasks widely encountered by knowledge workers. These tasks are generally believed to benefit from large displays that let users exploit spatial cognition to organize their findings. Additionally, inexpensive hardware makes large, tiled displays even more attractive for visual analysis and collaborative investigation. Especially in multi-user environments, the increased space allows to better organize the findings and results and, therefore, helps to improve collaboration. One important requirement is that all users can navigate seamlessly on the whole display space, while employing the standard software they are familiar with. In this thesis, we present a seamless desktop infrastructure for distributed cognition and collaboration. Our infrastructure only uses standard hardware and software. By choosing a minimally invasive, web-centric approach, we can integrate existing web applications and visual analysis software with little or no effort into our system. We can even leverage existing synchronization mechanisms built into many web applications today.

Moreover, we present an approach that enables individual and multiple analysts to organize concepts and relationship extracted from web documents in an observation graph, and, show how deep visual links to the exact phrases in the original documents and bidirectional link traversal improve the sensemaking process. In multiple studies, we explore how the availability of a sensemaking tool influences users' knowledge externalization strategies.

Furthermore, content on modern computer screens is displayed within the boundaries of windows. In many cases, however, not all content relevant for a task is immediately accessible to a user: it can be distributed over multiple windows that can occlude each other or can be minimized. Also, not all content fits into the viewport of a window so that parts of the content are "scrolled away". In search tasks, the efficient retrieval of such content is important. Current software, however, only provides limited support to visualize occurrences of entries outside of the current viewport within individual

applications, and rarely supports crossing application boundaries. For this reason, we introduce novel visualization methods to guide users to such hidden content. We show the validity of our methods in a user study which demonstrates that our technique enables a faster localization of hidden content compared to traditional search functions and thereby assists users in information retrieval tasks.

Contents

1 Introduction

In modern information analysis, investigators need to tie together many diverse pieces of information from multiple sources. When analyzing multiple large documents (using a broad definition of document to include any data source) not all information can be kept in plain sight. Content on computer screens is often inaccessible to users because it is *hidden*, e.g., occluded by other windows, outside the viewport, or *overlooked* even though it is technically visible [BDB06]. In search tasks, the efficient retrieval of sought content is important. Current software, however, only provides limited support to visualize hidden occurrences and rarely supports search synchronization crossing application boundaries.

Furthermore, extracting facts out of document collections (*information foraging*) and synthesizing knowledge (*sensemaking*) are tasks widely encountered by knowledge workers. Academics, for example, read papers for a general understanding, but also to extract specific facts and quotations. Intelligence analysts browse documents to identify and synthesize relevant pieces of information, such as the actors or the methods used in a drug trafficking operation.

Large displays, such as the one in Figure 1.1 provide the necessary space to let multiple users simultaneously inspect multiple documents and make comparisons without having to switch windows. Thus, any user can chose to focus on a specific part of the display, the "work zone" [AN12]. The remaining display space serves as a cache for revisiting the information later. Users can organize information by placing documents in dedicated locations without interfering with each other. Piling topically related documents resembles handling paper documents [AEN10]. Another benefit of using physical space for sensemaking is that raw document collections can be used without meta-information or preprocessing.

However, large displays also have disadvantages. They are more expensive, require more maintenance, consume more power and complicate software development. Moreover, designing effective interaction techniques can be challenging [Ni+06]. These limitations

Figure 1.1: Our largest tiled display consists of 24 monitors (50″, 1080p), arranged in a 8×3 grid and driven by a cluster of eight computers.

suggest that large displays are not the ultimate sensemaking solution. In fact, recent research has identified diminishing returns in terms of performance improvements with increased display space for text documents [Lis+15].

Dedicated document analysis software (or "sensemaking tools") [Wri+06; SGL07] represent an alternative to large displays. Such analysis software packages often support visual exploration of documents through a combination of natural language processing with information synthesis (e.g., by building a mind-map). Some tools [MT14] let the user *explicitly structure* the *observations* in a graph with *concepts* as nodes and *relationships* as edges.

1.1 Contributions

In the course of this thesis, we developed a sensemaking framework for multiple users that combines a large display with an explicit observation graph (OG). Furthermore, we introduced novel visualization methods to guide users to hidden content. The resulting system has several novel aspects:

- A *seamless desktop* connects multiple users, displays, computers, input devices and document windows.

- A *minimally invasive* software architecture requires only a browser plugin and a background service process.
- An *observation graph* facilitates the collection of concepts and their relationships in a compact, explicit manner.
- *Visual links to hidden content* guide to occluded or hidden content and improve awareness of related, visible content.
- *Deep visual links* connect observations directly with evidence inside documents scattered across the desktop.
- *Bidirectional link traversal* allows recalling observations from evidence, and evidence from observations.

We discuss the technical implementation of our system and report on explorations of how individual users and user pairs perform sensemaking tasks when they are given a choice of implicit and explicit structuring of information.

1.2 Publications and Collaboration Statement

The following publications (in chronological order) were published in the course of this thesis and serve as foundation for the thesis text:

- Thomas Geymayer, Markus Steinberger, Alexander Lex, Marc Streit, and Dieter Schmalstieg. "Show me the Invisible: Visualizing Hidden Content." In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. ACM, 2014 [Gey+14]
  Contributions
  - Visualizing hidden content (out-of-viewport content, occluded content, off-screen content (minimized windows, unopened documents))
  - Improved links bundling and reduced clutter
  - Exploratory user study

  Thomas Geymayer was the main creator of the concepts and system implementation, and, execution of the user study. Markus Steinberger supported the implementation and provided a GPU based routing algorithm. All authors contributed to the concepts, design and evaluation of the study and writing of the paper.

- Thomas Geymayer and Dieter Schmalstieg. "Collaborative Distributed Cognition Using A Seamless Desktop Infrastructure." In: *Proceedings of the IEEE Virtual Reality Workshop on Immersive Analytics (IA)*. IEEE, 2016 [GS16]
  Contributions
  - Multiple sets of input devices (multiples users) can operate on one computer, each with it's own mouse cursor. Each application has the input focus of on set of input devices.
  - Operate multiple computers with one set of input devices. Users can move with their mouse cursor and connected input devices seamlessly across all connected computers.
  - Clone web browser instances across computers (with optional synchronizing the views in both instances)
  - Peer-to-peer communication framework (input forwarding and JSON protocol)
  - Glass-sheet on top of desktop to render visualizations (eg. visual links)
  - Visual Links across computers

  Thomas Geymayer was the main creator of the concepts and system implementation. Dieter Schmalstieg contributed to the concepts and writing of the paper. Mark Dokter, Manuela Waldner, Alexander Lex, Marc Streit and Markus Steinberger provided valuable feedback for the concept and implementation.

- Thomas Geymayer, Manuela Waldner, Alexander Lex, and Dieter Schmalstieg. "How Sensemaking Tools Influence Display Space Usage." In: *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2017 [Gey+17]
  Contributions
  - Concept Graph with deep linking and automatic placement of windows around the graph.
  - Study on the influence of using the concept graph on the users display space usage.

  Thomas Geymayer was the main creator of the concepts and system implementation, and, execution of the user study. Manuela Waldner contributed significantly to the design of the user study. All authors contributed to the concepts, the evaluation of the study and writing of the paper.

Furthermore, Daniel Brajko needs to be explicitely mentioned because he selected, set up and maintained most of the hardware, notably a tiled display consisting of 24 monitors.

2 Related Work

2.1 Searching Information

As mentioned in chapter 1, the two main reasons for available content not being considered is that it is *hidden*, i.e., not visible to the user, or that it is *overlooked* even though it is technically visible [BDB06]. We first discuss techniques to show and access hidden content, followed by techniques that help avoid overlooking content.

2.1.1 Hidden Content

There are several causes for information being *hidden* from a user:

- **Large documents.** Many documents contain more content than can sensibly be displayed at a time, making it necessary to show only a small fraction of the data contained. We refer to the visible section of a document as its viewport, which, in order to read the whole document, must be changed through scrolling and zooming.
- **Occluding windows.** Frequently, many application windows are open, and unrelated content is covering important information, making it harder for a user to locate relevant information.
- **Minimized windows.** To reduce the number of open applications, users regularly minimize windows. Minimized windows may contain important information which is not present on the screen.
- **Closed documents.** Relevant information may be contained in unopened or not accessed documents.

For hidden content, we distinguish between techniques that reveal occluded or out-of-viewport content that could be shown within the given display space and techniques that visualize off-screen content.

A common strategy to reveal **occluded content** are see-through interfaces that cull away the foreground to reveal the background. A prominent example are magic lenses [Bie+93], which can not only remove foreground, but also alter the representation in arbitrary ways. Magic lenses are typically invoked manually and locally (only the elements within a 'lens' are changed), a characteristic that is not suitable for search tasks.

While completely transparent windows [Har+95] and user interface elements [HKV95] have been evaluated in the past, in practice they are rarely used when reading and interacting with content in window managers. An approach that uses transparency for 'unimportant' window regions is described by Ishak and Feiner [IF04], while Baudisch and Gutwin introduce 'multiblending' [BG04], as a smarter alternative to alpha blending that considers multiple image features and makes the blending results more readable.

A more sophisticated approach is taken by Waldner et al. [Wal+11b], who superimpose 'clip-outs' of occluded content on top of occluding components. They use a measure of salience to reveal salient occluded content in regions where the occluding elements are not salient. Steinberger et al. [SWS12] additionally scale unimportant content to reveal occluded content. None of these approaches use a semantic measure of relevance, i.e., consider what is currently relevant to a user. Search tasks, however, inherently require revealing specific semantic elements.

A common strategy to visualize **off-screen** content are marks rendered within the visible area that point to off-screen content. Baudisch and Rosenholz introduce *halos* [BR03], circles with the center at the (off-screen) point of interest and a radius chosen so that a segment of the circle intersects with the display. Similar in spirit are *wedges* [Gus+08], a technique that uses triangle instead of circles. Users can infer the location of the point from the size and position of the circle or triangle segment. A problem of halos and, to a lesser extent of wedges, is scalability and clutter if many targets should be indicated. To address this, Waldner et al. [Wal+10] introduce arrows that aggregate multiple proximate points of interest and indicate how many items are aggregated with the size of the arrow.

Highly abstracted document previews are another technique to visualize off-screen content. Eick and Ball use such abstract representations for visualizing software [ESS92]. Hearst combines them with a visualization of the search term density in specific regions of text [Hea95]. This technique has been extended by Dieberger and Russel [DR02] to consider multiple search terms and enable fast navigation and preview. The occurrences

of search terms can also be superimposed on the scroll bar [Byr99], a technique that is nowadays, for example, employed in web browsers and in software development tools.

However, all of these tools and techniques use very abstract representations. We believe that using a representation that preserves the appearance of the document under inspection will provide additional benefit to users. In our work we combine both marks to indicate the presence of off-screen context and on-demand smart previews to minimize clutter and enable efficient discovery of and navigation to off-screen content.

2.1.2 Overlooked content

Especially on large displays it is also possible to overlook content, even though it is technically visible.

Search combined with highlighting is a common method to minimize overlooking of content. The highlighting typically employs adding a colored frame, but other methods such as magnifying search terms [Suh+02] or interesting parts while shrinking [HF01; HF03] or even completely removing other content is possible [Bau+04]. Such methods are also used for generally 'interesting' content [HF01]. Stoffel et al. use a similar approach for thumbnail previews, where they distort important terms, while preserving the layout of the document [Sto+12].

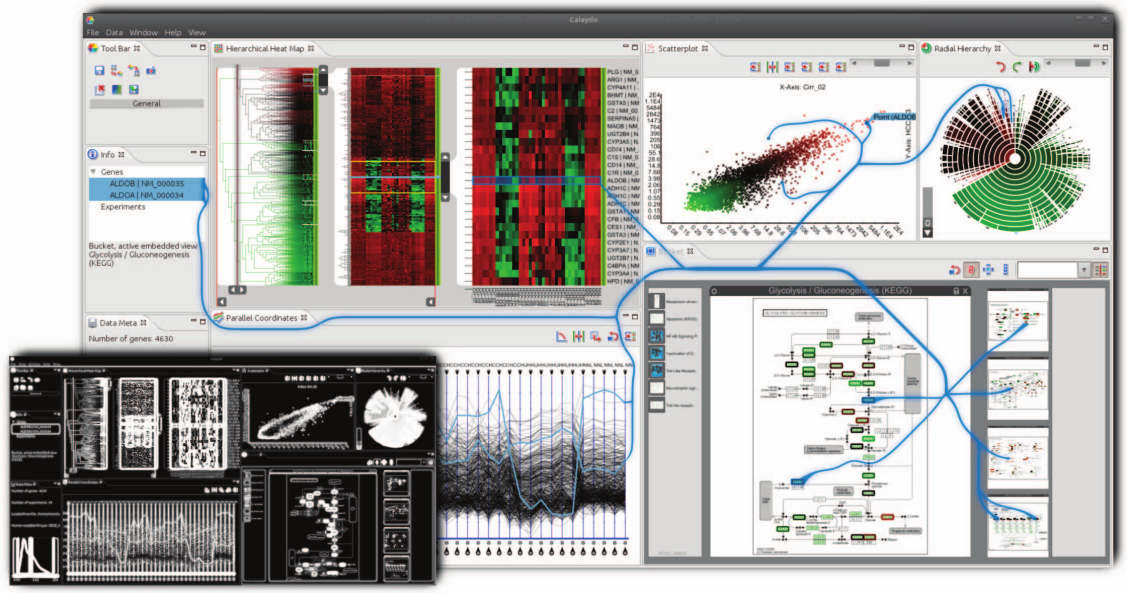


Figure 2.1: Context-preserving visual links [Ste+11] connect related entities while avoiding salient display regions.

A different approach is to employ connectedness [PR94], i.e., visual links [CC07], as a method to highlight occurrences of related elements (e.g., text, map locations or elements in a graph) inside application windows (Figure 2.1), which are beneficial especially in a cluttered environment such as in an information visualization system [CC07; Str+08; Str+09; Wal+10; VM12] or on large screens where not all content is in a user's field of view [HBW08; WS11; CN18]. Hoffman et al. [HBW08] have evaluated various techniques to help users locate windows on large displays and found that links (trails) outperform conventional highlighting (frames); similar results have been found for conventional displays [Ste+11].

We believe that visual links are the most powerful of these methods, especially in a highly heterogeneous environment spanning multiple windows and various types of documents, and consequently have chosen to combine them with methods to visualize hidden content in this work.

2.2 Sensemaking on the Desktop

There is a wide variety of work on large displays and tools for sensemaking of unstructured data, from which we draw inspiration. We loosely organize this body of related work into externalization tools, spatial organization, and collaborative uses.

2.2.1 Externalization Tools

Professional analysts employ document analysis software ("sensemaking tools"). This type of software often supports the exploration of documents through external cognition, which combines internal and external representations to perform cognitive tasks [SR96]. Externalization of one's internal knowledge reduces the internal memory load and supports cognition by being able to directly perceive the information [Kir95; Zha97]. Mind maps, concept maps, and similar visual representations of knowledge are commonly used externalization strategies, as is the spatial organization of work artifacts [Mal83; Kid94]. Indeed, Goyal et al. [GLF13] showed that users' sensemaking performance improved significantly when provided with a visualization of shared entities across documents compared to when only provided with a note-taking tool.

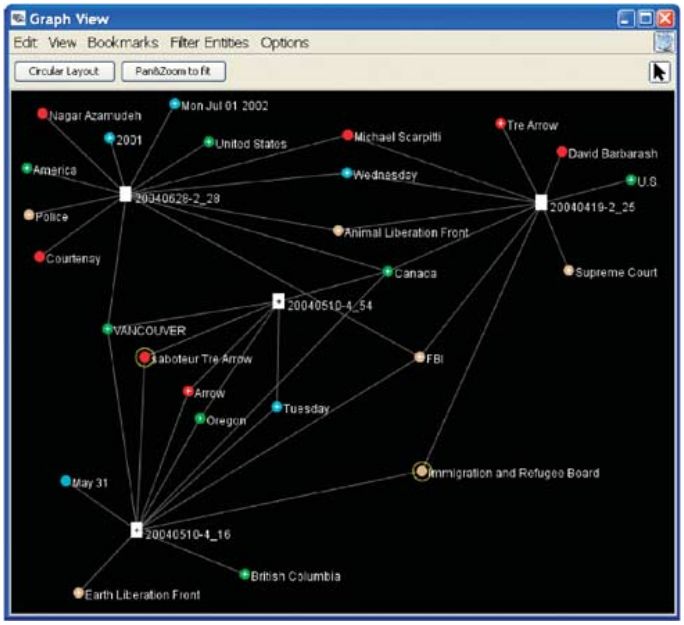


Figure 2.2: In *Jigsaw*'s [SGL07] graph view, documents are represented by white rectangles and entities by colored circles. These entities are automatically extracted and shown on demand.

Besides commercial knowledge management and analysis applications, which are often not scientifically documented, there are multiple important academic tools. *Jigsaw* [SGL07] is a text analytics tool to visualize extracted text entities in different representations, such as graphs (Figure 2.2), scatter plots, or fairly free-form "shoeboxes", in which the connections between entities are made explicit by color coding and visual links. Goyal et al. [GLF13] showed that users' sensemaking performance improved significantly when provided with a visualization of shared entities across documents compared to when only provided with a note-taking tool. The *Analyst's Workspace* [AN12] provides similar capabilities as *Jigsaw*, but exploits a vast amount of display space for externalizing the human memory by spatial organization of entities. Note that all of these examples employ pre-processing of a well-defined dataset. Ideally, a general-purpose sensemaking tool should not require pre-processing to allow for spontaneous integration of various kinds of data sources and formats.

Indeed, document analysis tools exist to help users understand an unstructured body of information. A powerful example is *nSpace*'s *Sandbox* [Wri+06], which supports information gathering from different sources and manual arrangement or automatic categorization of these information artifacts to externalize discovered facts. Similarly, *CLIP* [MT14] enables users to structure their knowledge in a graph, where entities are represented as nodes, which can be connected by edges and linked to a timeline. In

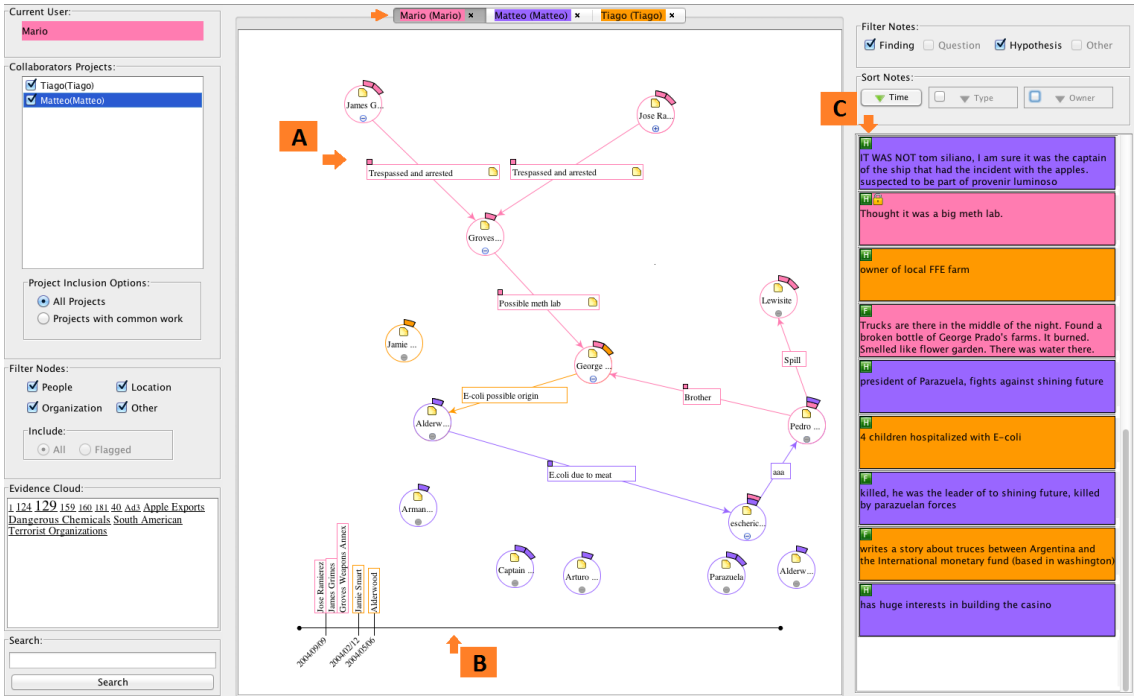


Figure 2.3: *CLIP* [MT14] allows to structure knowledge in a graph (A), as well as in a timeline (B). Furthermore note taking (C) is supported.

addition, users can attach an "evidence list" of relevant documents to nodes, as shown in Figure 2.3. In *ScratchPad* [Goto7], nodes can represent entire website snapshots, elements (like text passages or images) on websites, or user-specified concepts, which can be interactively linked. Similar tools have been introduced to create collections or structured summaries of web site components [Sch+02; Don+06] or information collections from arbitrary sources [Ber+08] in a pre-defined layout. Kandogan et al. [Kan+11] showed that free-form spatial interfaces are perceived as more flexible and provide a potentially clearer perceptual arrangement than strictly laid out interfaces, like a spreadsheet. Some sophisticated commercial note-taking software (e.g., *EverNote*[1] or *OneNote*[2]) support collecting users' notes and information snippets, like screenshots or web pages. Similarly, several mind mapping tools (e.g., *VUE* [SK05]) enable users not only to add text nodes, but also images and document links to the mind map for externalizing their mental concepts.

[1] https://evernote.com/
[2] https://www.onenote.com/

2.2.2 Spatial Organization

As an alternative approach to complex sensemaking tools, large displays provide "space to think" [AEN10], which can support analysts as well. In absence of other tools, information foraging and sensemaking is facilitated through spatial organization of information in documents and relationship extraction from multiple documents [AN12].

Multiple researchers have found improved performance in analysis tasks when using large displays compared to small displays [Cze+03; BNB07; YHN07; Liu+14; Red+15], and documented increased subjective satisfaction [AEN10; BB09]. However, it has recently been shown that performance does not increase linearly with increasing display space [Lis+15] and that effective interaction can be challenging [Ni+06]. Jakobsen and Hornbaek [RH11] compared the use of focus+context, overview+detail, and zooming techniques on three different display sizes. In their study, performance on a very large display setup was not superior to a medium-sized display.

Among the opportunities of large displays are the ability to divide the space into focus and context areas [Gru01; BB09], placing application windows as reminders [HS04], as well as clustering or piling windows [AEN10; Wal+11a]. Large displays thereby act as *externalized memory*, as users employ the space to organize and memorize information [AEN10]. These spatial organization strategies have not only been observed on large displays when working with digital data, but also on physical desks, where knowledge workers often use piling to organize their papers [Mal83; Kid94]. To support spatial organization strategies on large displays, researchers have introduced novel spatial document and window management techniques. For instance, *Scalable Fabric* [Rob+04] supports the categorization into focus and context areas by scaling down peripheral windows on a moderately large display space. The *Analyst's Workspace* enables users to manage piles and other spatial organizations of window groups and uses visual links to connect related entities between document windows [AN12]. *Collaborative information linking* combines a large single-display groupware with a bookmarking tool and visual links to let users solve a knowledge-intensive task [WS11]. *VisPorter* [Chu+14] combines spatial document arrangement with a collaborative concept map, while *Savil* [CN18], as shown in Figure 2.4, uses visual links on documents directly.

Figure 2.4: Sensemaking with SAViL [CN18] using text documents and visual links on multiple displays.

2.2.3 Collaborative use of large displays

Several special-purpose applications have been developed specifically with collaboration support in mind. These information systems include topics such as web search [AM08; TDR09], text analysis [IF09], collaborative picture galleries [MPW06] and network visualization [Ise+09]. For example, in a collaborative tabletop environment, *Cambiera* [IF09] supports the spatial arrangement and mutual awareness of opened documents and keyword searches within this documents. However, no abstraction into concepts, relations or (visual) links to specific sections within documents are possible.

Tools supporting co-located, synchronous user interactions do not need to be built from scratch. Forlines et al. [For+06] built a wrapper for Google Earth to support viewport synchronization, multi-user interaction, and annotations, without changing the application's core implementation. Isenberg et al. [Ise+09] are following a similar approach, by, for example, adding multiple color-coded mouse cursors within an existing applications. Both tools still need modifications to support collaboration and therefore can not easily be combined with other groupware applications.

To overcome these limitations in conventional windowing systems, Hutterer and Thomas [HT08] created multi-pointer X (MPX), an extension to the X Window System for multi-pointer

interaction. This is in line with the observations of Lauwers and Lantz [LL90] which have already suggested using existing single-user applications in any windowing system with collaboration support. However, this approach does not support any more advanced collaboration features, such as interaction histories, access control mechanisms or any automated translation or communication between applications.

Adding multi-input support to a single, shared display is commonly referred to as a single-display groupware (SDG) [SBD99]. Typical examples for SDG systems are wall displays [CBF14; Ise+09] or tabletops [IF09; MPW06; TIC09; TDR09]. One problem reported commonly is the mutual distraction of users while performing individual work. For example, cursor movements by other users [Wal+09] and changes of the spatial display layout [EGR91] are often causes for disruption. As a result multiple, users often use their personal territories on a high-resolution display to visualize a lot of information simultaneously [Ise+09; TDR09; Bra+13]. This can lead to fragmentation and loss of mutual awareness [SS97]. On large displays, common problems include locating the cursor and items of interest within the whole information space [SS97] and interacting with distant display locations [RL04]. Spotlights [Kha+05] have been introduced to guide the users attention to individual regions on large screens. VisLink [CC07] and visual links across applications [Wal+10; CN18] use visual line connections to guide users to related elements (e.g., text or map locations) inside distinct application windows as well as outside the users visible field of view. Visual links have also been shown useful in multi-user SDG systems to show relations between information in personal windows and shared information [WS11].

Bradel et al. [Bra+13] investigated collaborative sensemaking on a large display using either Jigsaw or a simple document viewer with highlighting and annotation. They observed that users had fewer documents open with Jigsaw compared to the document viewer, but speculated that this difference was caused by the different window management behaviors of the two sensemaking tools. However, an alternative explanation could be that the users employed different externalization strategies in Jigsaw, so that the actual *need* for multiple document windows was reduced.

Currently many visual analytics applications are adopting web technologies. To handle multiple devices and users, several frameworks allow application developers distributing their web applications across multiple displays. An early example is Multibrowsing [Joh+01] which allows moving information to different devices connected to the system. More recent systems, like Panelrama [YW14] and PolyChrome [BE14], allow

splitting the user interface across multiple devices, but require a deeper integration with the individual applications. SAGE2 [Mar+14] and Savil [CN18] cover the entire large display with web-browser windows and provide a multi-user interaction capable windowing system running in the browser, at the cost of no longer being able to run native applications.

2.3 Enabling technologies

In this section, we describe technologies, software and frameworks that are essential to the implementation of our system:

Qt[3] is a cross-platform application development framework, which we use in many parts of our system: We create different types of windows, including a transparent, full-screen, "glass-sheet" window, used to draw visualizations on the desktop, small, moving windows to render custom cursors, and, normal windows with ordinary GUI elements. There are components for networking, including WebSockets, parsing and creating of JSON encoded documents, and, launching and controlling of external applications. Furthermore, we use code from the (no longer maintained) **libqxt**[4], which is an extension library for Qt. It provides cross-platform, system-wide, window related operations, for example, get information about all opened windows on a computer (e.g., position and size on the screen and window title), change the state of arbitrary windows of other applications (e.g., minimize, maximize, resize, and, acquire focus) and, register global keyboard shortcuts.

The **X Window System**[5] provides on many, especially Linux-based, operating systems the basic framework and protocol to display, move and resize windows, and, interact with them using mouse, keyboard and other input devices. Today it is slowly replaced by Wayland[6]. In our work on multi-user systems we strongly depend on the Multi-Pointer X [HT08] (MPX) extension to the X server, which allows to use multiple, independent input devices on one computer.

Synergy[7] is an input redirection system, which initially was completely open source,

[3] https://www.qt.io/
[4] http://bitbucket.org/libqxt/libqxt
[5] https://www.x.org
[6] https://wayland.freedesktop.org/
[7] https://symless.com/synergy

but current development is only partially released as open source[8]. We based our work on the original, open source version of Synergy, modified by Dokter [Dok10] to use MPX for allowing multiple users controlling multiple computers simultaneously.

With the **Firefox**[9] web browser, we integrate the web as application platform. We use it to show local files as well as resources and application on the web. **D3.js**[10] is a JavaScript library to create dynamic and interactive *D*ata-*D*riven *D*ocuments, which we used to create a fully interactive sensemaking tool (section 3.3). **Recoll**[11], a full-text search tool operating in the web browser, we use to provide searching in local files on our system.

[8] https://github.com/symless/synergy-core
[9] https://www.mozilla.org/firefox/
[10] https://d3js.org/
[11] http://www.recoll.org/

3 Sensemaking Environment

In this chapter, we describe the sensemaking environment that we have designed during the course of this thesis. A large *seamless desktop* lets users employ familiar tools for foraging by providing a "shoebox" for potentially interesting information. *Visual links* together with *guidance to hidden content* help users in finding and locating information. An *observation graph* lets the users structure findings explicitly at any time without disrupting the workflow. Users can connect observations with evidence using *deep visual links* to maintain an overview of their work. Finally, the visual links allow for *bidirectional traversal*, so that users can conveniently navigate without fearing to become lost in the data. An additional benefit of our sensemaking environment is that it lends itself to collaborative work, as we will show later.

3.1 Requirements and Design Principles

No investigation or analysis task is the same. To allow for usage of our techniques in a broad field of scenarios and independent of office space setup and budget, we formulated several requirements and principles used while designing and developing our system:

Commodity Hardware To allow for a broad usage, standard computer hardware and operating systems should be supported. Every reasonably new desktop and laptop computer should be able to run our system.

Scalability Users should be able to take advantage of all available hardware. This includes connecting multiple computers to the system, as well as using multiple or larger (tiled) displays. Furthermore computers and components should be able to join or leave the system dynamically at any time, without influencing other parts of the system.

Minimally Invasive Instead of providing a single monolithic and specialized analysis software, a software platform should be provided which is compatible with a wide

range of existing tools, platforms and work practices. Existing applications should not require any or only small modifications and continue to work normally, even if they are used outside of the software platform. Standard behaviors of the operating system should be supported, like drag and drop, clipboard and other interaction techniques.

Modularity Existing workflows should be enhanced instead of enforcing new workflows. Users should be able to choose which parts of the system to use and which parts are not required or should be replaced by different software.

Open Standards and Open Source To increase interoperability with existing and new applications, open standards should be used whenever available and suitable. Using open source technologies increases longevity and simplifies customization and adaptation to specific environments.

During the course of this thesis, we performed experiments in a small office for up to three persons, and, a larger laboratory with a tiled display. In the office, we used a standard desktop computer, a standard laptop computer and a desktop computer with six 22-inch monitors connected as a small tiled display, as shown in Figure 3.1. In the laboratory, we worked with a tiled display of 24 50-inch monitors (Figure 1.1), where either 16 monitors could be driven by a single computer or 24 monitors by a cluster of 8 computers. All computers where either connected using ethernet or Wi-Fi.

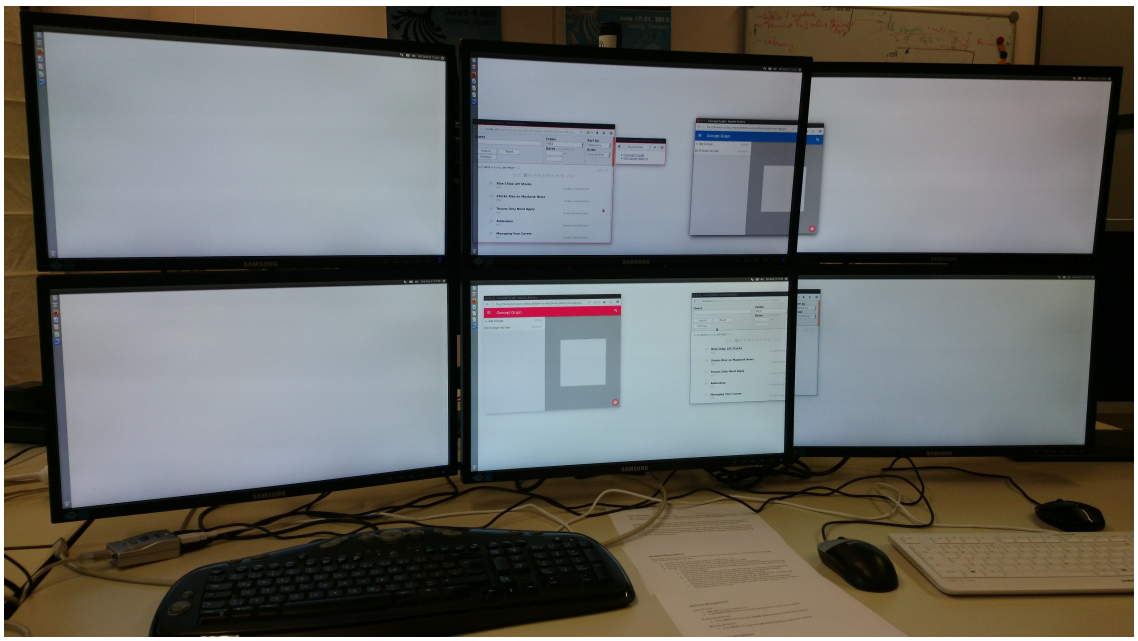


Figure 3.1: Six 22-inch monitors are setup to for a tiled display. Two pairs of keyboard and mouse allow two users to interact with it simultaneously.

3.2 Seamless Collaborative Desktop

Figure 3.2: Example of an arrangement of two standard computers (left & right) and one computer with a large (six monitor) display (center) on an office table. Users can "teleport" with their mouse to each computer without changing the actual physical input devices.

We wanted a seamless desktop that is agnostic of the computing infrastructure, be it a single host computer, or a cluster of hosts [GS16]. The entire seamless desktop should be opportunistically controlled by multiple users, simply by grabbing any free mouse/keyboard. Additional computers, such as notebooks, can be added using standard cabled or wireless network infrastructure. Adding users should be possible at any time, even in the middle of a session.

What is required to successfully convey the illusion that users are interacting together on a seamless desktop? The first group of requirements concerns the users' input devices. We assume one mouse and one keyboard per user. Users should be able to use their input devices as usual, without having to consider display boundaries, as shown in Figure 3.2, or coordinate with other users. This implies the following requirements:

- **R1** Multiple computers should be operated with one set of input devices.
- **R2** Multiple sets of input devices should be usable on one computer, including multiple mouse cursors.
- **R3** Multiple application foci should be provided, so that multiple users can interact one-on-one with applications on a single computer.

The second group of requirements concerns the applications:

- **R4** Multiple applications should run concurrently, either on the same or on different computers.

R5 Multiple clones of one application window can be created and linked to the state (i. e., viewport scrolling) of the master window.

R6 Multiple instances of a document, which are being modified by multiple users, should be automatically synchronized.

3.2.1 Design

To allow for a *minimally invasive* approach that extends, rather than replaces, existing software applications, we selected a standard web-browser (Firefox on Linux) as the main application platform. The minimally invasive approach is realized by a only requiring a web-browser plugin and a background service for the operating system (Figure 3.3). Both use only the standard API provided by operating system and browser. All standard features of operating system and browser remain accessible at any time, and native software can be used as well. Existing applications do not require any modification and continue to work normally, even if they are disconnected from the special sensemaking infrastructure. This approach differs from previous approaches, which usually wrap all the physical computing resources into a monolithic software framework. Letting users keep their familiar tools benefits productivity and satisfaction.

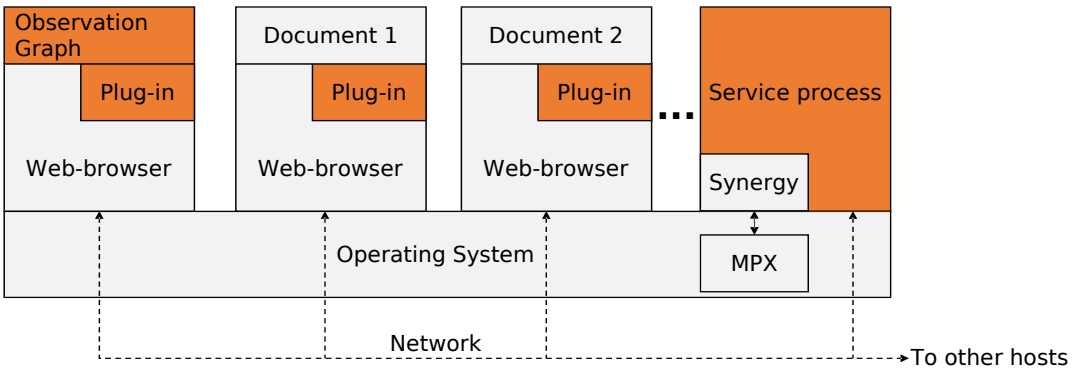


Figure 3.3: The software architecture is minimally invasive: It only requires the installation of a web-browser plug-in and a service process. The observation graph runs as a web application. New software components are shown in orange.

On each host, a service process must be started, with the first host running as a server and all other hosts as clients. The server host advertises its address via periodic broadcasts. Client service processes listen to the broadcasts and connect to the advertised server address. After establishing the connections, the server brokers communication among clients using a JSON protocol, which we will describe in detail in section 4.1.1.

To facilitate a seamless desktop, the arrangement of the displays on each client host is recorded in a spatial model maintained by the server. In addition to the spatial model, the server maintains a list of user records, indicating the host where the user's input devices are connected and assigning a distinct cursor color for each user. To display visualizations and user interface elements for collaboration, it opens a transparent overlay window covering the entire desktop. A control GUI can be invoked on demand to interactively set these parameters.

3.2.2 Input redirection

Our system provides *bidirectional input redirection* from any physical input device to any display in the cluster (**R1**). Events from a physical input device are intercepted and injected into the event manager of the destination host. On the destination computer, the input events appear to be coming from virtual input devices. We use a heavily modified version of Synergy[1], based on the work of Mark Dokter [Dok10], which works bidirectionally: On the one hand, every host injects events from the attached input devices into the network, tagged with a particular user's identity. On the other hand, every host receives events from all other host and determines if they concern the part of the seamless desktop managed on that host, based on the spatial model.

Moreover, we use MPX to support *multiple sets of input devices* per host [HT08] (**R2**). MPX supports *multiple concurrent window foci* on our main application platform, Firefox. We use the transparent overlay to render multiple cursors in distinct colors. By connecting MPX to our input redirection, *any user may control any application on any display connected to any computer using any input device* (**R3**). With ample display space, joint inspection and manipulation of the same document can be carried out in two synchronized windows, as we will explain in the next section.

3.2.3 Web application redirection

Support for multiple concurrent applications on one computer (**R4**) is trivially fulfilled by a multi-tasking system such as Linux. Each computer in the cluster runs arbitrary native applications, so that users can keep using their favorite tools.

[1] http://synergy-project.org/

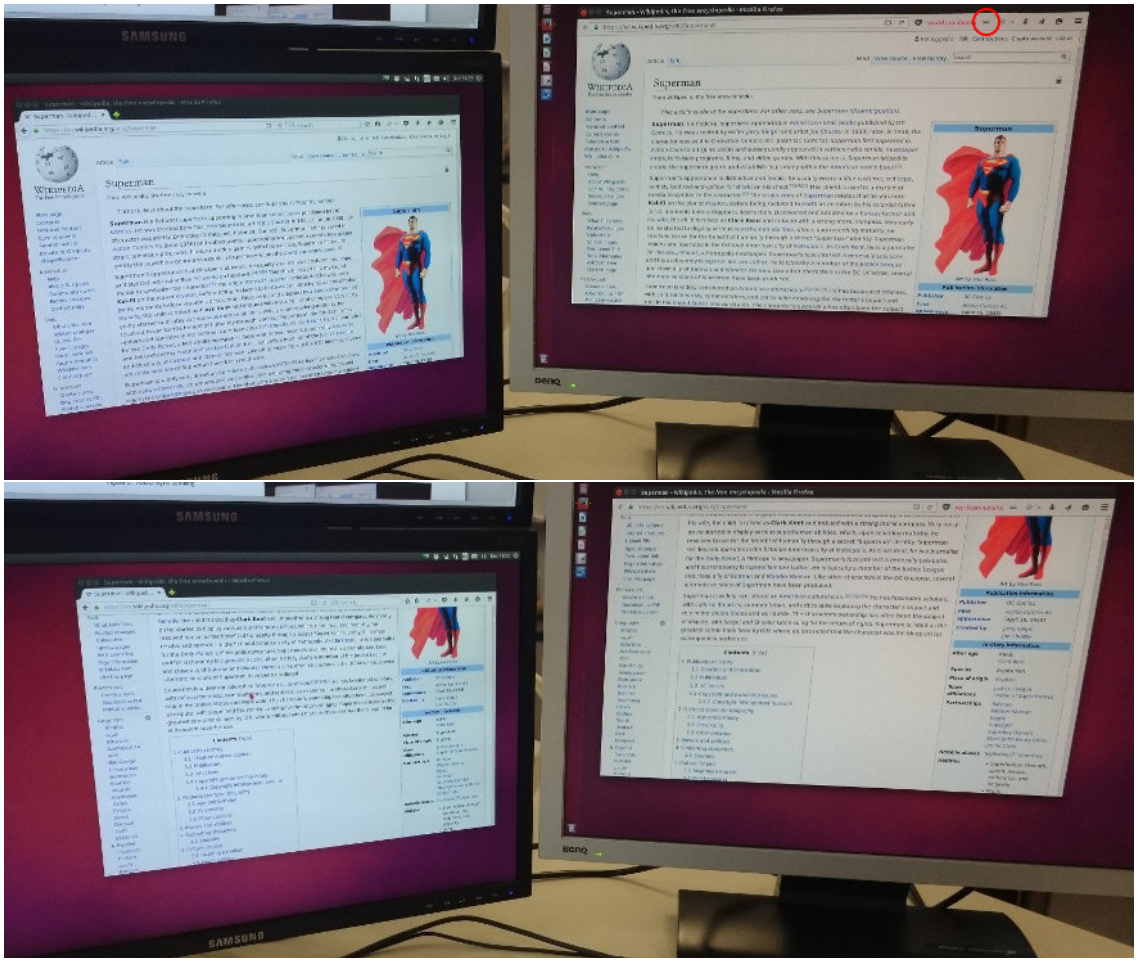


Figure 3.4: Cloning a web-browser to another desktop. An icon in the browsers toolbar indicates that it is synchronized to another browser (red circle in the left image). After moving the viewport in the source browser (left), the viewport of the synchronized browser (right) has automatically been updated.

However, our main application platform is the web-browser. It puts an additional level of indirection between applications and the operating system. For example, applications and data in web applications are loaded on demand, avoiding the need to install applications before use. This makes it easy to migrate application state across computers.

Conventional static web pages and server-side technologies such as REST encode the user's document view in the URL. Feeding this URL into a web-browser on another computer results in displaying a clone of the same web page, that is a document view with the same content. We exploit this fact by letting the user invoke a cloned web-browser instance with the same URL on any computer in the cluster. The clone can be coupled to the master instance (Figure 3.4), so that that basic browsing functions (scrolling, resizing, scaling, page change) are reflected by the clone (**R5**). This is useful

for "teleporting" information across large display areas to make them better readable for a collaborating user [BBo5]. We also allow to reserve the roles of master and clone or enable bi-directional synchronization, as long as only one user at a time manipulates the browser.

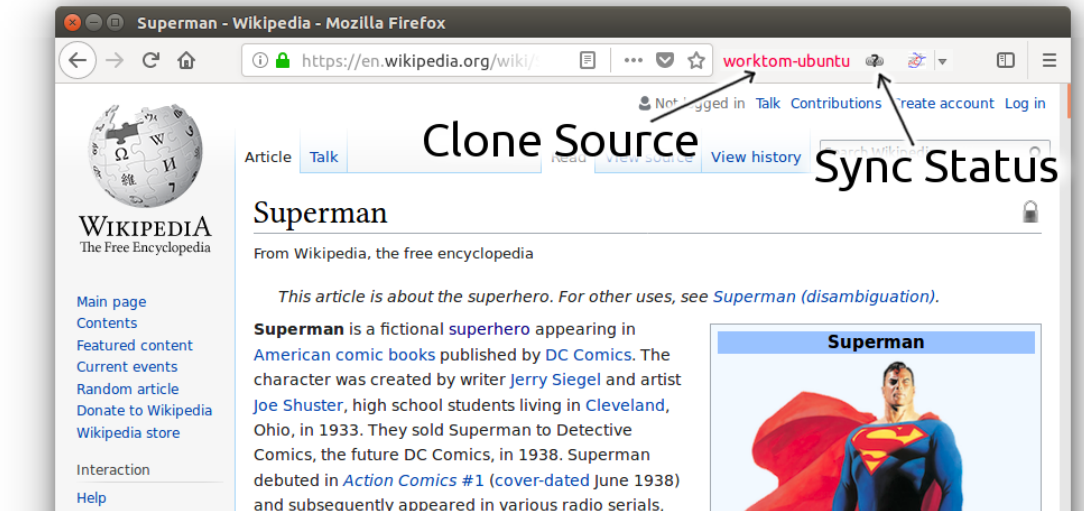


Figure 3.5: In the cloned window there are two additional elements available in the browsers toolbar. One indicates the source of the clone, and the second one the connection status.

Additional controls in the browser toolbar (Figure 3.5) indicate the state of the connection and allow to change the synchronization mode. The source of the cloned window is indicated by showing the name of the computer and the color of the user of the source window. A button with an icon provides information about the status of the synchronization, which can be *unknown*, *synchronized*, or *disconnected*. When the connection to the source window has been lost, or before receiving any updates from the source window, the status is unknown. Otherwise a click on the button enables or disables synchronization, effectively changing the roles and mode of synchronization.

Combining input redirection and web-browser cloning achieves the illusion of a seamless desktop. For instance, a user can *drag a web-browser window across multiple displays even if they are connected to different hosts*. What really happens is that the original window is closed and a clone of it is created at the destination host. During dragging, a captured image of the browser content is shown on the overlay window.

That is, while traversing the source node's desktop, normal dragging is carried out by the local window manager. As soon as the mouse moves to the target node' desktop, an event in the input redirection component is triggered. A preview picture of the

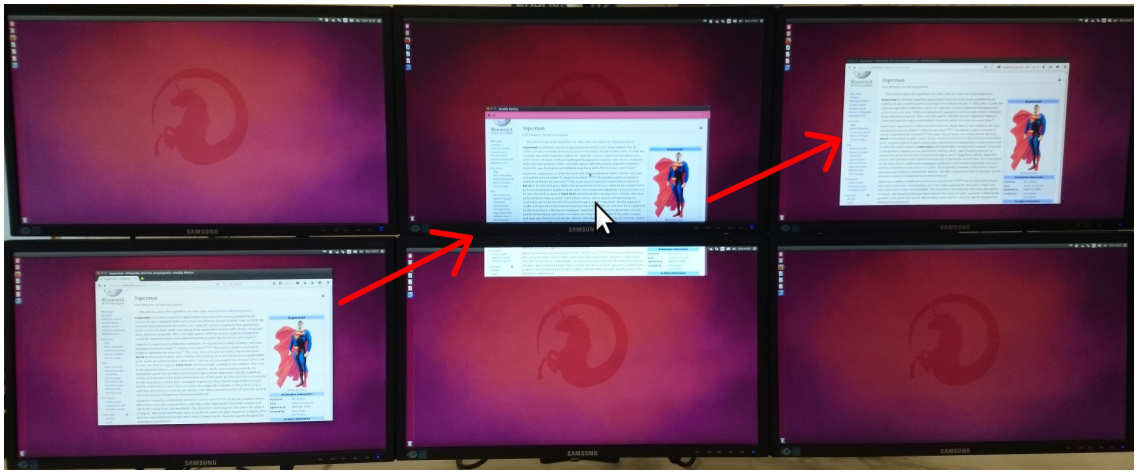


Figure 3.6: Migrating a browser window to another desktop. The source window is shown on the left, a preview while dragging in the middle, and the target window on the right.

web-browser window is rendered and transmitted to the daemon on the target node, along with the current URL of the dragged web-browser (Figure 3.6). At the target node, the preview picture is attached to the mouse cursor and rendered on the glass-sheet. When the user drops the web-browser in the target location, the daemon on the target note creates a new web-browser instance with the URL specified in the input redirection event. In a final step, the source web-browser is deleted to finish the illusion that the web-browser has been migrated to the new location. Alternatively, a *copy operation* retains both web-browser instances and synchronizes their behavior.

Since we allow arbitrary web applications, document manipulations across browser instances are not automatically synchronized. However, many web applications are able to synchronize with their web server in real-time. Changes reported to the web server are forwarded to other users viewing the same document, effectively allowing shared work (**R6**). One could serialize the full internal state of a master web-browser and duplicate it in a clone. However, for the analysis scenarios investigated so far, we found it sufficient to rely on cloning the URL and using shared web applications such as Google Docs or Wikipedia.

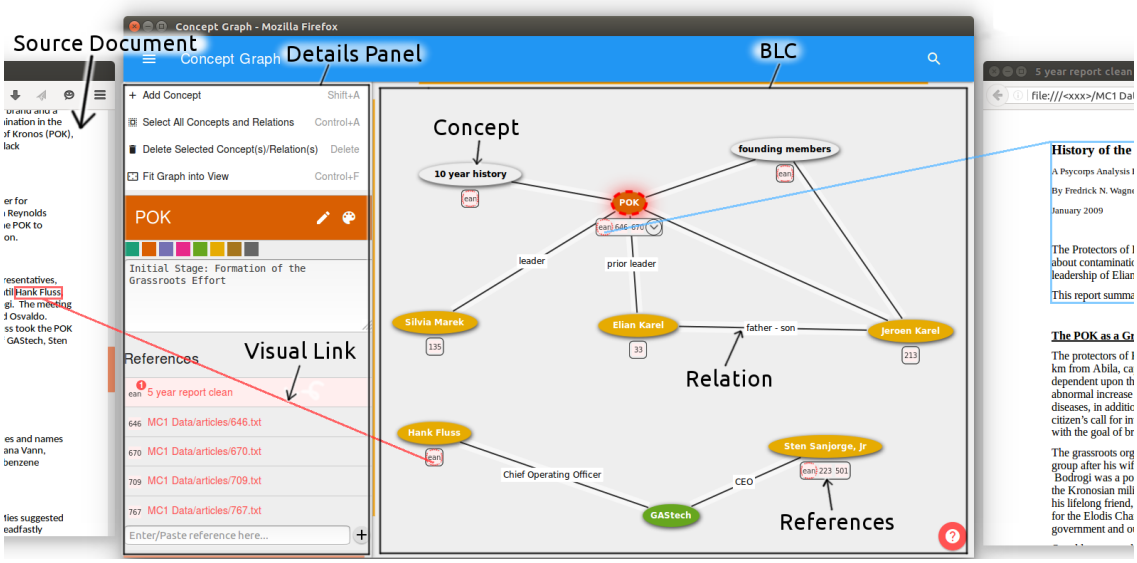


Figure 3.7: OG and two source documents. In OG, concepts are rendered as nodes, relationships as edges. Here, the concept "POK" is selected and its properties are shown in the detail panel on the left. References that act as evidence for a node are shown as small circles below the nodes. The dashed red frame around the reference circle and the red circle showing "1" in the reference list indicate that the document is currently open. The reference for the node "Hank Fluss" is opened on the left, and the reference for the node "POK" on the right. The specific text sections that support the nodes are highlighted and connected with visual links to the reference circles.

3.3 Observation Graph

With the observation graph (OG) we introduce a lightweight, general-purpose sense-making tool that supports users in organizing their findings into observations (concepts and relationships), displayed as a node-edge diagram. The nodes are laid out manually, which is meant to act as memory aid and as an external representation of the user's internal knowledge [Kir95], which can help to make inferences [LS87]. Observations can be color-coded, for example, to classify nodes. Every observation can be given a unique name and can be associated with additional data, such as textual notes. Details about a selected observation are provided on demand in a side-panel, as shown for the node "POK" on the left side of the OG in Figure 3.7. In larger observation graphs, OG supports selection of individual observations, rubberbanding, zooming, panning, and searching for observation by name. A video demonstrating the usage of the OG is available online[2].

OG lets users link each observation to multiple pieces of evidence from the source documents, serving as evidence for its validity. Links can be created either manually or

[2] https://youtu.be/mDbbrWxaRC0

through keyword search in documents. The system treats the "external" links created in this way as fist class objects by making them *deep* and *bidirectional*.

Links to evidence are deep: Links can not only refer to entire documents, but inside these documents also to individual phrases or terms as well as non-textual content like images or graphs. Deep links allow a user to quickly revisit the exact piece of evidence they were previously investigating. When document views are open, cross-application visual links [Gey+14] are used to connect observations with evidence.

Link traversal is bidirectional: Users can revisit evidence from observations, or they can revisit observations from evidence. Hence, users can freely arrange documents on a large display, with or without attributing meaning to the placement. They can chose to keep the documents open or close them after visitation without the fear of being unable to find the exact evidence again in a large pile of documents. When users reopen documents, the system automatically places the document window close to the referenced graph node. Hence, users can make effective use of the large display space to organize their information sources without considerable window management overhead.

Taken together, these two properties of links make working with many documents significantly more powerful, in particular, on large displays. Our OG implementation is aware of all currently open and active documents. When a document is activated, all links to it are highlighted in the OG. This enables analysts to quickly identify how important it is with respect to the overall information captured in the the OG.

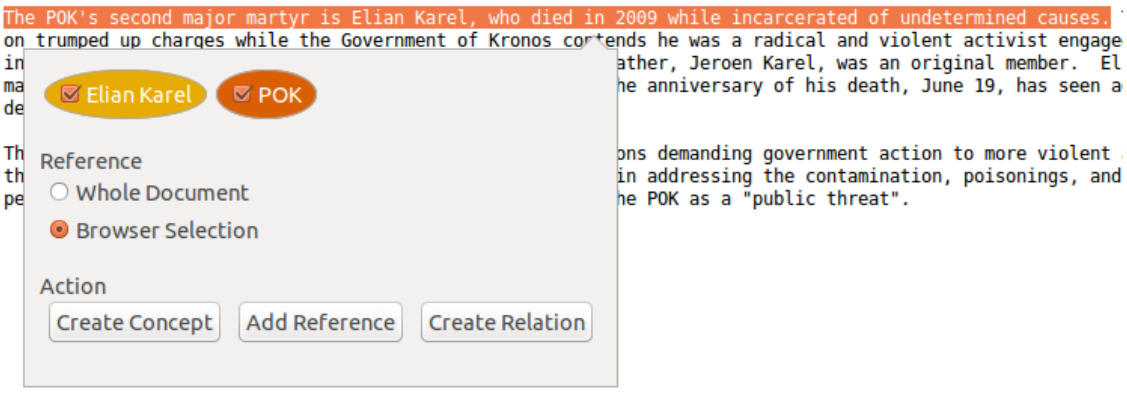
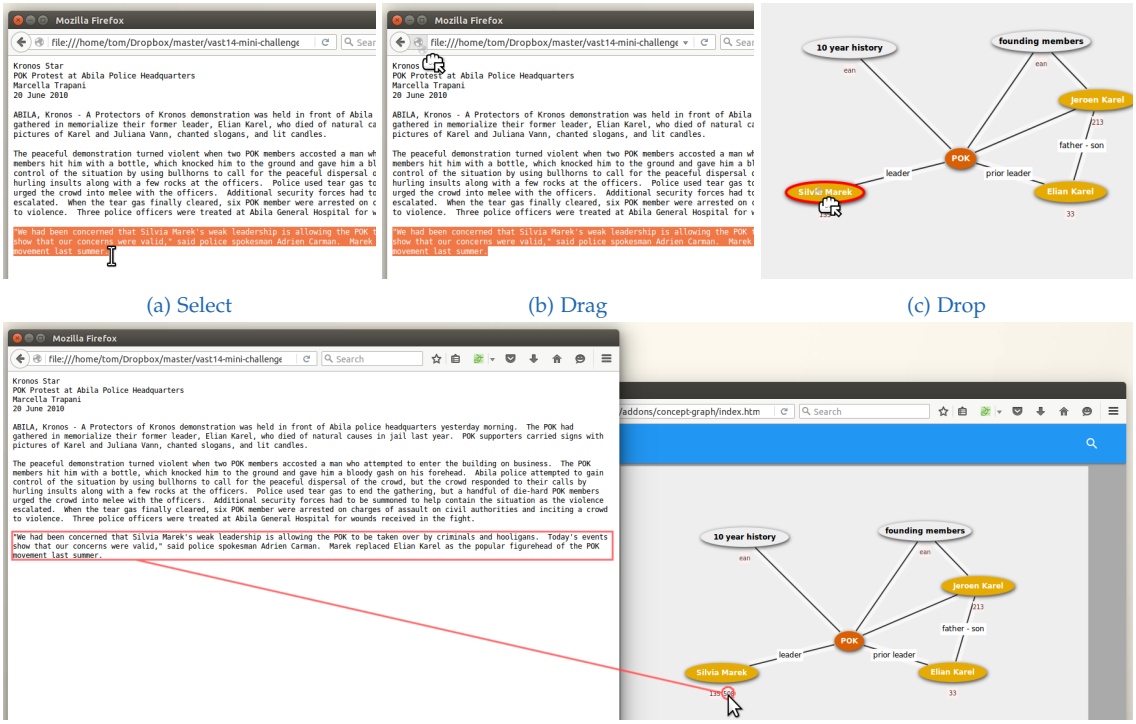


Figure 3.8: Adding a relationship to two concepts. A highlighted section of a document can be added as a new relationship between nodes using a context menu. This can be used to create new concepts, to connect selected concepts, or to add references to concepts or relationships. In this case, a reference is added between "Elian Karel" and "POK".

3.3.1 Interaction with the observation graph

Observations can be either created manually in the OG, or directly in a source document, based on mouse selection. When creating concepts out of a source document, the deep link to the selected statement inside the document is automatically attached to the node or edge. Nodes and edges out of source documents can either be created from the context menu of a selection within a document, as shown in Figure 3.8, or by dragging the selection onto the OG viewer, as shown in Figure 3.9. When dropping a selection into the OG viewer onto a node or edge, the reference is added to the according concept or relationship. Otherwise, a new node is added at the position of the drop.



(a) Select (b) Drag (c) Drop

(d) Revisit

Figure 3.9: Adding a reference with drag and drop. If desired a specific section within the document can be selected ((a)). Then the selection or the icon next to the browsers address bar can be dragged ((b)), and dropped onto the graph ((c)). Clicking on a reference opens the according document and initiates a visual link to the referenced content ((d))

Users can open multiple OG instances, zooming in on specific observations. Multiple OG instances are helpful if concurrently working users want to specialize or if the OG is too far away from the currently inspected document. Each user can own a personal OG instance, indicated by drawing the graphs header in the user's personal color. The owner of a graph is always the last user who interacted with that graph instance.

Each OG instance has individual navigation (viewport, zoom-level, search terms), and an individual selection of observations. The selected observations are used when the user invokes a context menu to add links to currently selected observations. For example, when the user selects two concepts in the OG and then opens the context menu, the user will be presented with the possibility to link the selected concepts with the current page (and, optionally, the selected phrase).

3.3.2 Deep visual links

In the OG, links are represented as small glyphs adjacent to the observations. When a link is selected, the referenced window is brought into focus. If no window is displaying the requested document, a new window is opened and placed as closely to the node as possible, which leads to a dynamic spatial organization, prioritizing the current working set of documents. Moreover, the document is automatically scrolled to the location of the evidence. The evidence itself is highlighted with a colored frame and connected to the graph with a visual link, as can be seen in Figure 3.9d. The reference glyphs attached to the nodes also reveal whether the document is currently open on the desktop (red highlight in Figure 3.10).

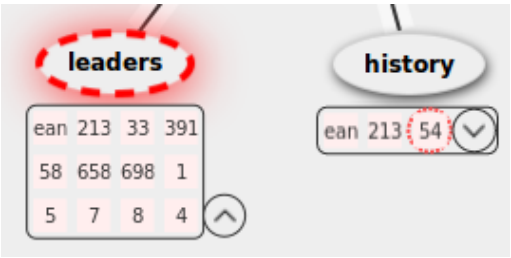


Figure 3.10: Reference glyphs in the Observation graph. Once a configurable maximum number of references is exceeded, they can be collapsed and expanded. A red circle (right) indicates that the according document is currently open on the desktop.

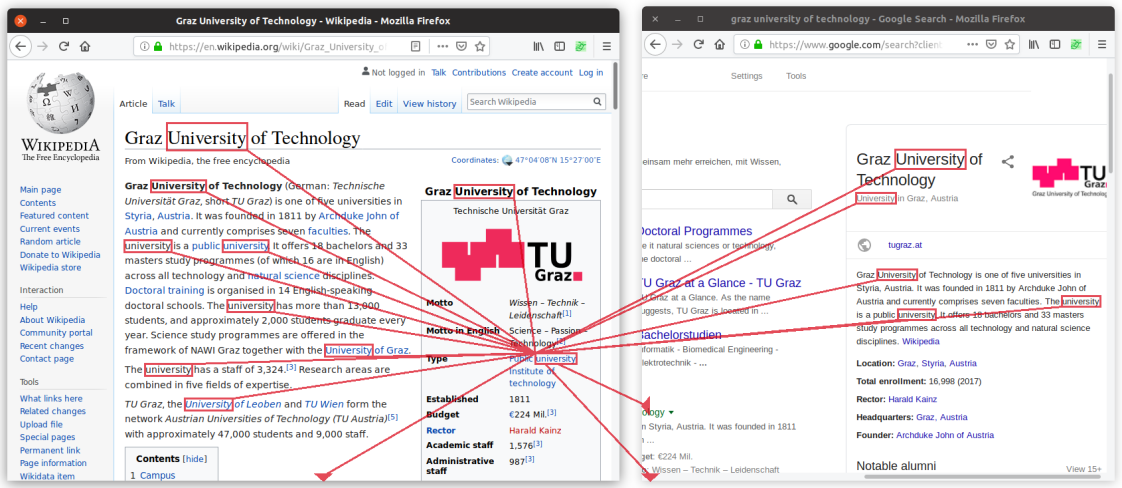
3.4 Visual Linking

As discussed in section 2.1.2, visible content can be overlooked for many reasons. While traditional highlighting techniques such as color work reasonably well on a uniform background, a cluttered environment, e.g., due to many windows, or other factors such as large display sizes can increase the chances of relevant content being overlooked [HBW08]. A remedy for overlooking content are visual links [Ste+11], explicit edges connecting individual pieces of information. The system developed in the course of this thesis [Gey+14] is based on previous work of the author [Gey13]. Compared to

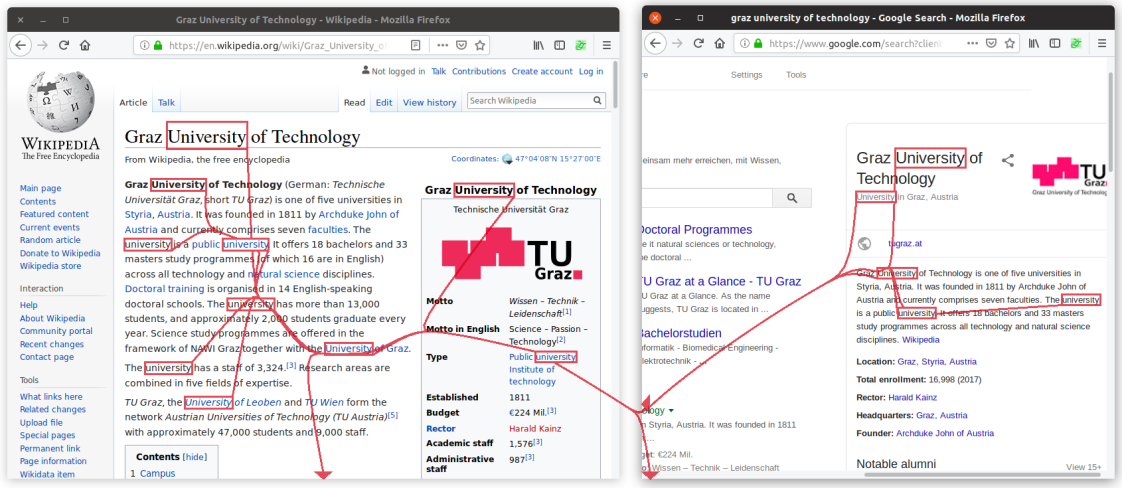
the previous system, we decreased visual clutter and improved usability and visual appearance.

After the service process has received a user-triggered search string, it forwards the request to every connected client application, enabling each application to add its regions. Upon receiving a request, each client searches its content for instances of the requested identifier and reports back the bounding boxes of all found occurrences. For simple selection types, like individual words, bounding boxes already provide an accurate approximation of the relevant region. To highlight and link more complex shapes, such as objects in a map or graph, a client is free to use arbitrary polygons for representing its regions.

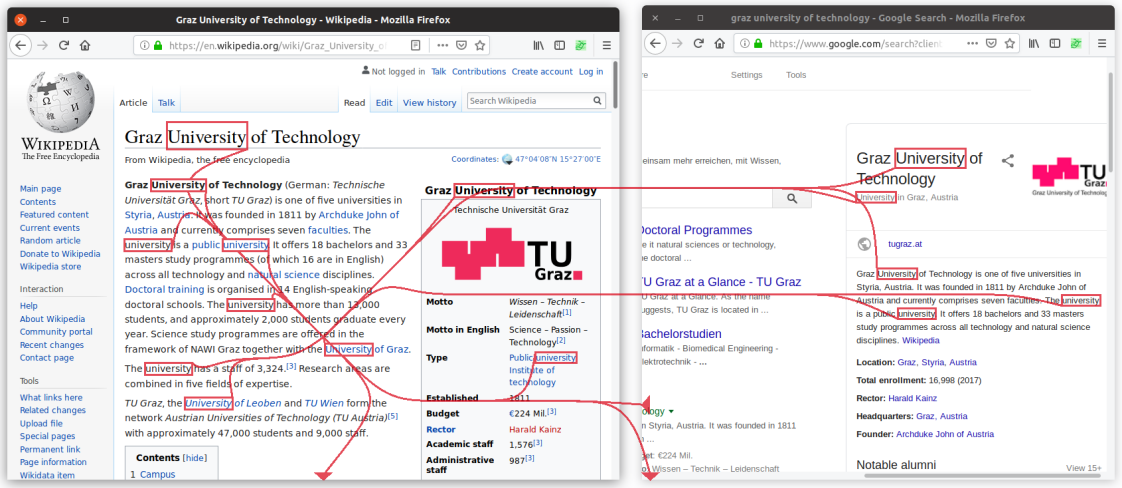
Visual links are drawn between all highlighted regions across application boundaries. The naive approach that connects all highlighted regions to a common center results in a cluttered visualization (Figure 3.11a). Therefore, we bundle links using force-directed edge bundling [HW09], an algorithm based on an iteratively refined system of control points, attracting each other (Figure 3.11b). The system is initialized by calculating the center of gravity of all occurrences. Then, the highlight region closest to the center of gravity is determined, and all other regions are connected to it. Moving the center of gravity avoids an artificial branching point. Next, all links are subdivided into segments of approximately equal length, and, finally, force-directed edge bundling is applied. Due to potentially large differences in the length of individual links, the forces affecting a single link can change rapidly, leading to sharp corners in the link routes. To address this issue, we apply a geometric smoothing on the points forming the link routes after executing the bundling algorithm.



(a) Straight routing



(b) Force-directed edge bundling [HW09]



(c) Dijkstra [Dij59] based routing

Figure 3.11: Comparison of straight routing of links to a common center (a), force-directed edge bundling (b) and a prototype of routing using Dijkstra's algorithm [Dij59] on a grid (c), which could be used to prioritize and penalize routing links in certain regions, as done by Steinberger et al. to prevent covering important content [Ste+11].

3.5 Visualizing Hidden Content

Directly accessing all content, both visible and hidden, poses a number of challenges. To address these challenges, we elicited six requirements that a technique for visualizing hidden content needs to address. This list evolved out of an initial set of requirements which we based on our experience with designing visualization interfaces for application domains such as systems biology and our own needs when working on search tasks. We then created multiple prototypes, which we informally evaluated with users, leading us to the following final set of requirements:

R I: Mental map. Users should be aided in building a mental map of the explored documents. Ideally, this overview should make use of the investigator's spatial memory by pointing out occurrences at their actual position on the desktop (if occluded), or at least preserve relative arrangement of occurrences (if currently not on the desktop). For such a mapping, it is necessary to communicate the information's *location* or at least the *direction* in which information can be found.

R II: Indicate occurrences and relevance. Sources or sections mentioning a search term multiple times tend to be more relevant compared to documents or parts that only contain a few instances of a term. Thus, users may want to prioritize densely populated documents or sections in their search. To support this requirement, visual cues should indicate the amount of information available at a specific location or in a certain direction.

R III: Fast previews. To enable users to quickly judge whether looking at a specific region in a document in detail could be interesting, getting fast previews of hidden content should be possible.

R IV: Fast navigation. Fast navigation between all available chunks of information is essential for an efficient exploration. Once a user has chosen to closely investigate a piece of hidden information, it should be possible to quickly navigate there with minimal interaction.

R V: Heterogeneous sources. The integration of information from different sources is essential to capture all types of hidden content. The technique should connect information from open application windows, minimized windows, and documents which are only present as a file on a disk or available on the Internet. All sources should be handled in a unified manner to allow for all types of hidden content.

R VI: Changing content. The technique must be able to accommodate arbitrary changes to the content presented on the desktop. In particular, the arrangement of windows

on the desktop can change at any time: Windows can be moved, resized, minimized, or closed. Moreover, the content and viewport of windows may change at any time, possibly removing existing information, adding new information, or changing location and visibility of information.

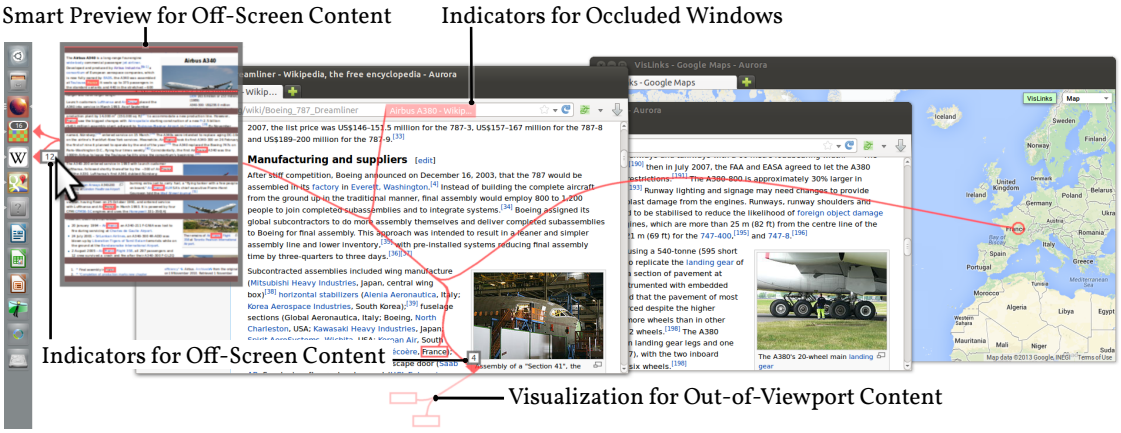


Figure 3.12: Visualization of occluded, out-of-viewport, and off-screen content. Multiple browser windows and files stored on the computer contain occurrences of the search term "France". Most occurrences are either outside one of the browser's viewports, which is indicated by arrows and links to the out-of-viewport locations, or are occluded by another window, which is indicated by semi-transparent red window labels. The arrows pointing to applications in the taskbar indicate off-screen content in a minimized window (Wikipedia) containing 12 occurrences of the search term and 16 files found by the desktop search engine. Hovering over an arrow, reveals a smart preview showing the regions of the documents containing occurrences.

For visualizing hidden content, we take two steps: indicating that relevant content is hidden in the first place, and revealing the hidden content on demand. We introduce novel techniques to visualize **occluded content**, i.e., content that is occluded by other windows, **out-of-viewport content**, i.e., content that is outside of the application window but within the limits of the display, and **off-screen content**, i.e., content that is outside of the available display space or that is in closed, minimized, or not accessed documents. All of these techniques are integrated with visual links to create a strong visual connection between all related pieces of information—hidden as well as visible—resulting in an information exploration interface that makes hidden content easily accessible and reduces the risk of overlooking content (see Figure 3.12). A video demonstrating our hidden content visualization techniques is available online[3].

[3] https://youtu.be/F2k4V8KGllI

3.5.1 Visualizing Out-of-Viewport Content

Relevant content is often hidden due to the limited viewport size. Large portions of content can be situated in virtual space outside of a window's viewport. We distinguish two cases of out-of-viewport content: (1) The target region is outside the current viewport, but would be visible on the screen if the application's viewport was extended, and (2) the target region is outside the screen. We consider the latter case as off-screen content, which is treated in a later section.

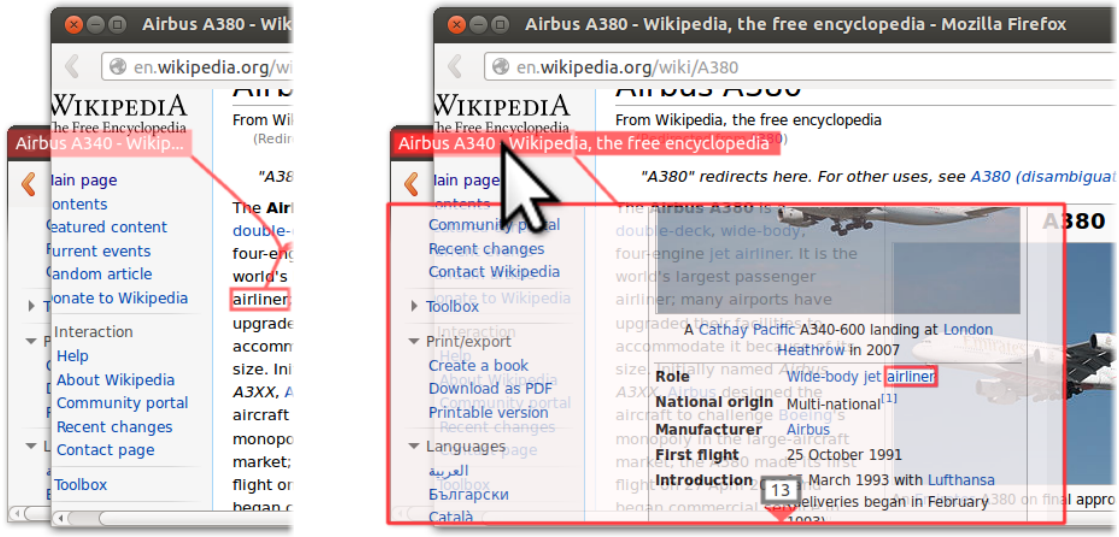
For the former case we use *smart links*: semi-transparent outlines—one for each target region—that indicate the location of invisible target regions (Figure 3.13a) and are connected to other visible or hidden occurrences with visual links. Smart links clearly indicate the occurrence and relevance of hidden content (R II) and support a user's mental map (R I). If the user hovers over a target region outside the application's viewport, as shown in Figure 3.13b, all of the application's content that fits on the desktop is rendered in order to provide context to the otherwise 'disembodied' target region, thus providing fast previews of the hidden content (R III). When the user selects such a target region, the application's viewport is automatically centered around the region, allowing the user to immediately continue working with the application at the chosen position, thus facilitating fast navigation (R IV).

(a) (b)

Figure 3.13: Links to out-of-viewport content. (a) Smart links pointing to out-of-viewport occurrences of a search term. (b) Hovering over a smart link reveals a semi-transparent overlay showing the actual content.

3.5.2 Visualizing Occluded Content

To direct the user's attention to windows containing occluded content, we use markers, connected to visual links, as shown in Figure 3.14a that contain the title of the windows where relevant content was found. We chose this approach over showing direct links to occluded content (as we do for out-of-viewport regions), since user-feedback indicated that direct links produce too much clutter. When hovering over such a marker, we overlay the hidden window semi-transparently and highlight and connect the relevant content (R III), as shown in Figure 3.14b. To avoid interference with the background window, the overlay can optionally be shown completely opaque. To ensure fast navigation to the occluded region (R IV), a click with the mouse on the overlay moves the respective window to the top of the window stack, thus permanently revealing the occluded information.



(a) (b)

Figure 3.14: See-through visualization for occluded content. (a) A marker showing a part of the window title indicates occluded content. (b) On hovering over the marker covered content is superimposed, including highlights for the occluded content.

3.5.3 Visualizing Off-Screen Content

If a target region is located either outside the screen, within a minimized windows, or in unopened files, it is not possible to draw a link to a specific target region. Instead, we visualize off-screen content by drawing an arrow pointing into the direction of the target, or at the icons representing the minimized windows and unopened files. To

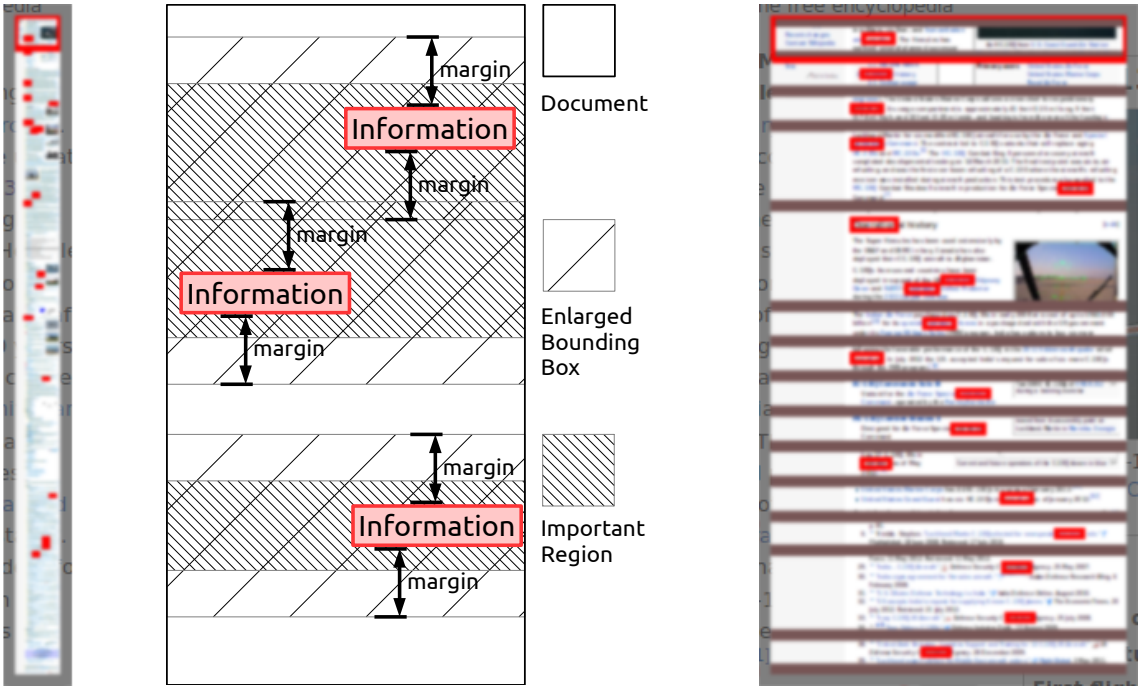


Figure 3.15: Smart preview. (a) A preview of a complete web page with the relevant regions highlighted. The necessary scaling makes it hard to recognize information. (b) All regions containing relevant information are detected and embedded within a bounding box, that makes sure some context is retained. Overlapping bounding boxes are merged. (c) By clipping the unimportant regions much more detail for the relevant parts is revealed.

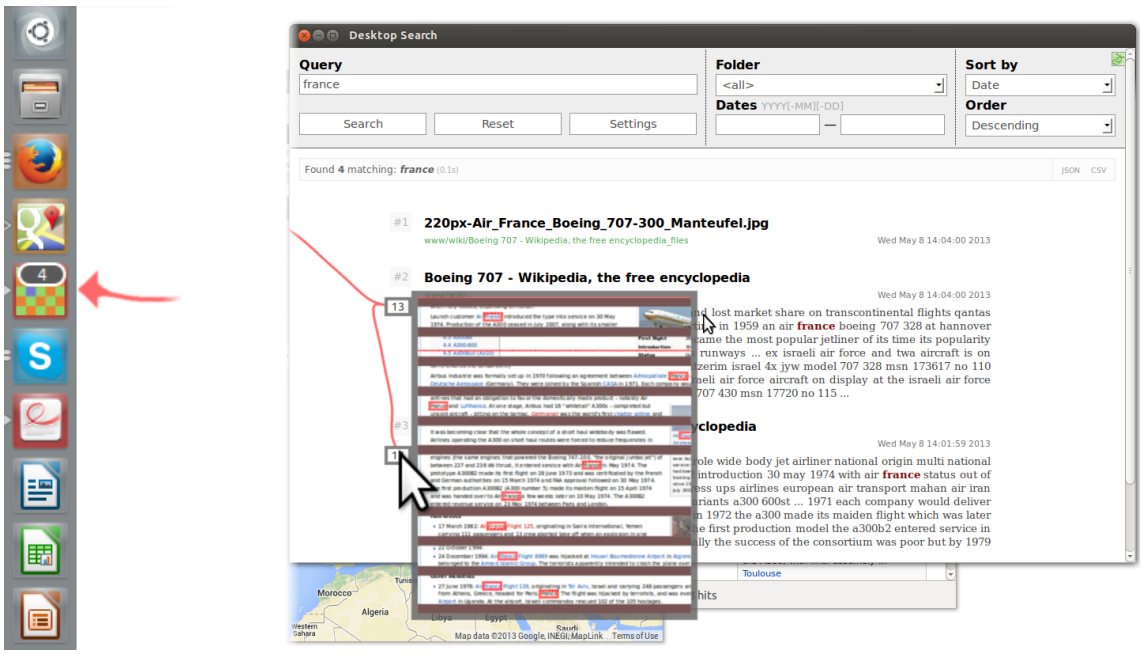
avoid clutter in cases where multiple target regions are off-screen, we only draw a single arrow for each icon or window edge. For the latter case we adjust the arrow to point towards the center of gravity of all outside regions. Additionally, we draw a text label next to the arrow to show the number of hidden target regions in the given direction (see Figure 3.12). These encodings indicate occurrences and relevance (R II).

To get an overview of the whole document (R I) and to enable fast navigation (R IV), we provide a *smart preview* of the complete document, which appears when hovering over the arrow. For search tasks, it is reasonable to assume that users are only interested in those parts of the document that contain relevant information. We use this consideration to present the user with a more compact preview where regions containing no relevant information are clipped, freeing up space for increasing the size of interesting areas (see Figure 3.15). In this way, all hidden regions of the document are presented at once. To decide which areas should be removed, we first calculate bounding boxes of all highlighted regions, then loop through all bounding boxes and mark regions with a certain margin above and below as important and finally hide all unmarked and therefore

unimportant regions (see Figure 3.15c).

These smart previews can be zoomed and panned to explore all target regions in detail (R III). To facilitate orientation (R I), the current viewport of the application is highlighted using a rectangle in the preview. Once a target region has been identified by the user, clicking on the target region hides the preview and scrolls the document to the location of the requested information (R IV).

When using the smart preview for **minimized windows**, we draw an arrow next to the application's icon in the task bar to indicate that it contains target regions. Upon hovering over the arrow, the smart preview is revealed (see Figure 3.12). When the user selects a target region, the minimized window is restored and the viewport is centered on the selected target.



(a) (b)

Figure 3.16: Links to unopened documents. (a) The icon of a desktop search engine is marked with an arrow. The number shown inside the icon indicates the number of documents containing the current search term. (b) Within the desktop search engine, occurrences are highlighted and smart previews are provided.

Information may also only be available in **unopened documents**. To integrate unopened documents, we query a desktop search engine to find occurrences of search terms. Similar to our approach for minimized applications, we draw an arrow next to the desktop search engine's icon in the task bar and show the number of documents found within the icon, as shown in Figure 3.16a. When opening the search engine, we highlight the

search term in the search engine's previews and provide smart previews to reveal the details, as shown in Figure 3.16b.

3.5.4 Collaborative Visual Links and Observation Graph

For efficient collaborative usage of Visual Links (section 3.4 and section 3.5) and OG (section 3.3) we identified two additional requirements:

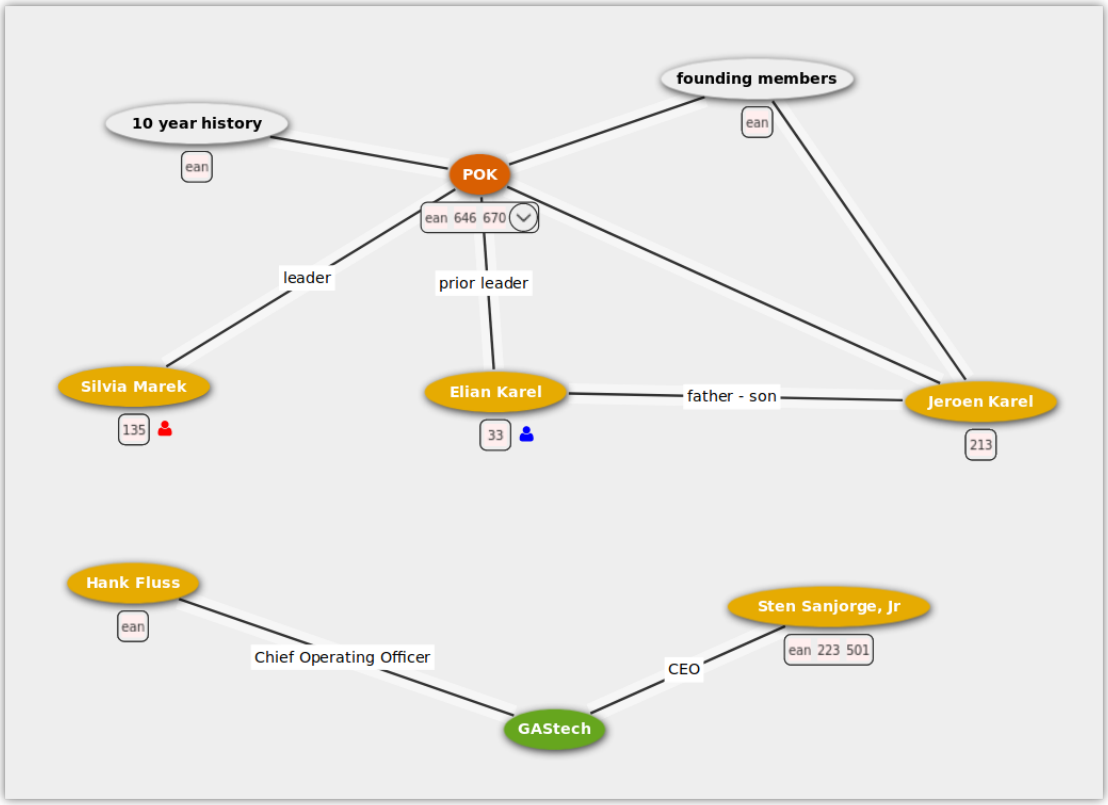


Figure 3.17: Observation graph with multiuser hints. Colored icons indicate search activity of other users. These icons appear next to concepts in the graph, whenever another users search or visual link matches the according concepts name. The color indicates the users identity and a click on the icon connects the own graph to the other users link.

First, users should be able to work concurrently, *without interfering* with each other. To reduce interference, visual links are assigned to users. Whenever a user invokes a new visual link, only already existing visual links created by that user are removed. Conversely, visual links from other users will not include a user's personal OG instance. This keeps interference low when performing individual search processes.

Second, users should be *aware* of available information and other user's search activities.

Awareness of visual link ownership is increased by assigning visual links a unique personal color per user. Besides, awareness of other users' search activities is retained within OG instances by placing icons next to a visual link, as shown in Figure 3.17. These icons indicate another user's activity related to a given observation. The color of the icon corresponds to said user's unique color. A click on the icon connects to the corresponding visual link and allows for a fast navigation to the other users regions of interest. This feature can be used to synchronize with other users about their findings.

During activation, visual links keep a list of owners. In the beginning, the list contains only the user which initiated the visual link. Whenever another user connects to an existing visual link, an entry is appended at the end of the list. Aborting or disconnecting from a visual link removes a user from the list. During activation, the visual link retains the color of the first user in the list. When the list is empty, the visual link is removed.

4 Implementation

In this chapter we will first describe the core of our system, which provides a powerful messaging infrastructure. Afterwards we will describe several components which add different capabilities to the system. Our infrastructure only uses standard hardware and software.

We observed that visual analysis software, like most types of information systems, is moving towards a web-centric architecture. Today, an analyst or knowledge worker wishing to conduct everyday work entirely using web applications can do so with relatively few restrictions. Consequently, we have chosen a minimally invasive, web-centric approach, to allow integrating existing web applications and visual analysis software with little or no effort into our system. We can even leverage existing synchronization mechanisms built into many web applications today.

Instead of building a monolithic framework, we *stitch* together individual desktop environments and software. This provides three important advantages: First, the engineering effort for the stitching is much smaller than development of a new monolithic framework, which would have to duplicate many existing GUI functions. Second, existing GUI functions, such as window management, do not have to be duplicated. Third, existing application software does not have to be re-written or ported to the new framework.

4.1 Overall Architecture and Communication Protocols

The backbone of our system is a central service written in C++, which runs in the background as a daemon and accepts connects from other applications taking part in our system. These other applications are integrated using a minimally invasive approach through the use of a plug-in API or minor modifications of the application source code, if no API is provided [Wal+10]. The data exchange between the server and the clients is handled using standard network sockets. We support Transmission Control Protocol

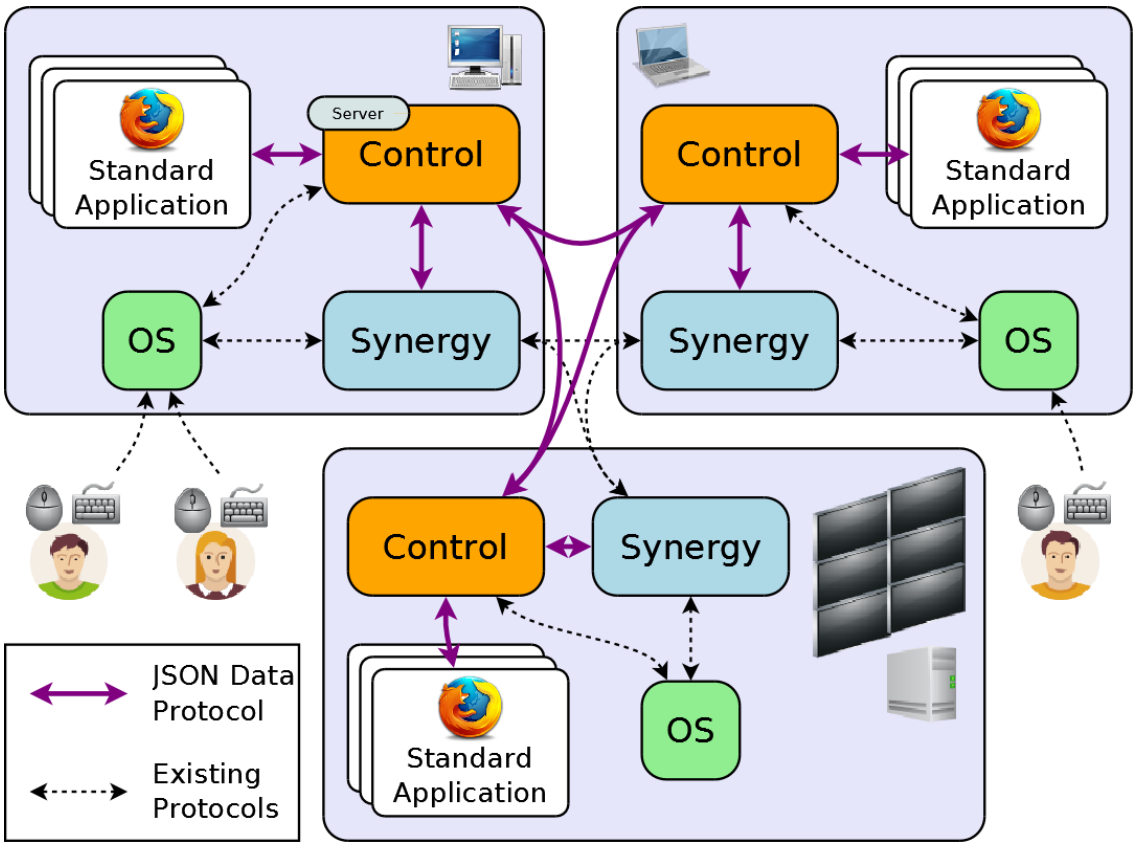


Figure 4.1: System overview with three computers connected to each other. Multiple users can control the same computer, each with their own set of input devices (see section 4.2.1). User input can be redirected to other computers (see section 3.2.2). Data is transferred with existing protocols as well as a custom, JSON based protocol (see section 4.1.1). Different client applications, for example a Browser (see section 4.1.3), can connect to the system.

(TCP) sockets, which are available in almost every programming language and hardware platform, and WebSockets [FM11], which are implemented on top of the TCP stack and have a good support in modern web browsers.

4.1.1 Protocol

We use a simple, yet powerful and flexible protocol. Messages are encoded as JavaScript Object Notation [Bra14] (JSON) objects, where every message is described by attributes of an object. A *'task'* property is required to identify the type of the message and any other property can be used to fit the requirements of each message type. An example message which is send upon registering a browser to our system is shown in Listing 4.1.

```
{
    'task': 'REGISTER',
    'type': 'Browser',
    'pid': 2957,
    'client-id': 'firefox:1482318266292:2957',
    'cmds': ['open-url', 'save-state', 'scroll'],
    'viewport': [137, 184, 296, 75],
    'geom': [135, 113, 300, 148],
    'url': 'https://www.tugraz.at/institute/icg/research/team-
        schmalstieg/'
}
```

Listing 4.1: Example message of a Firefox Browser instance registering to the system.

Several message types are known and handled by the server, but it also allows to forward unknown messages to another client. This enables customized client-to-client communication without any awareness or modification of the server required. We will discuss some of the supported message types later, when they are required by different parts of our system. Refer to the appendix for a complete list of commands.

Binary data can be sent as Base64 [Jos06] encoded string attributes of the JSON messages. Nevertheless, sometimes it is desirable to send raw binary data. For example, during the development of a browser extension (section 4.1.3) using the WebSocket API provoked noticeable delays, if the message size was too large, or even sometimes exceeded the maximum allowed size of text messages. To work around this problems and to improve efficiency, binary data is sent as raw binary data, whenever the underlying socket API supports it. Binary messages start with a header containing a message type and an ID or sequence number, followed by any amount of binary data. We use binary messages as primary means of transfer for (large) image data.

4.1.2 Multicomputer, Configuration & GUI

The server can also connect to servers on other computers and forward messages to them. To enable for an easy setup in a local network, every server broadcasts its name and port using the User Datagram Protocol (UDP) on a fixed port. A Qt-based GUI dialog (Figure 4.2) shows all servers for which it received a broadcast. By clicking on the name of a server a connection is established. Messages can be forwarded to a specific client on any connected computer, to all clients on a specific computer and broadcasted to all connected clients on all connected computers. The resulting architecture is depicted in Figure 4.1.

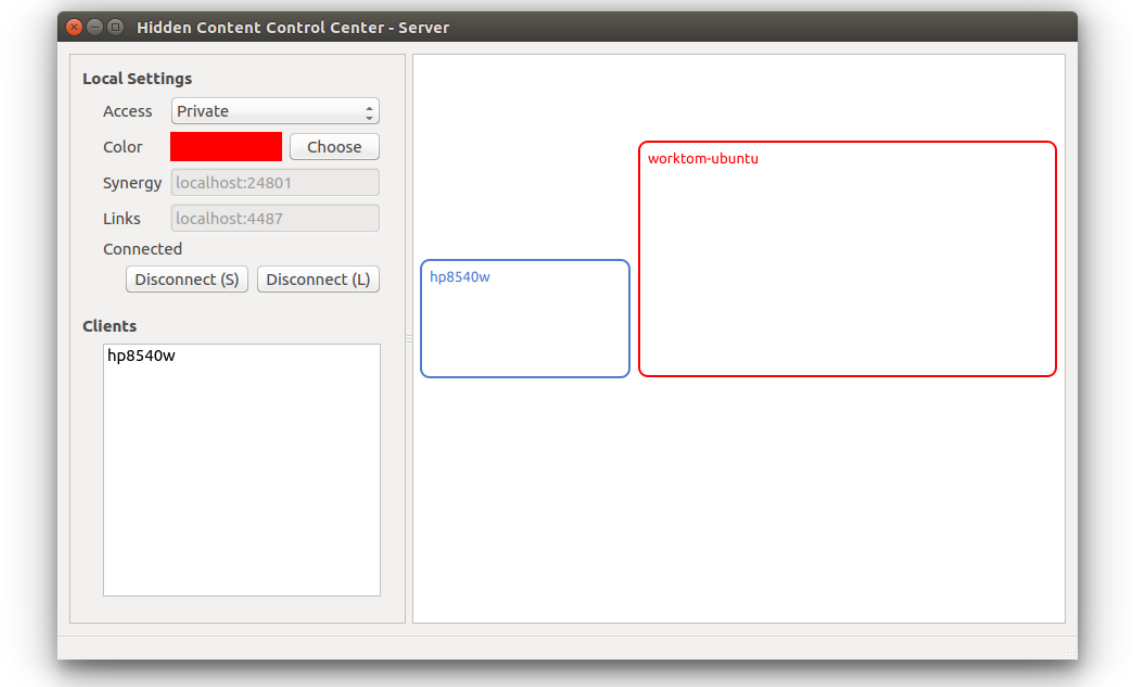


Figure 4.2: Configuration tool for setups with multiple computers. On the right side the screen spaces of all connected computers are aranged. Here two computers are connected: One large screen (red) and a smaller laptop (blue).

The largest part of the GUI is taken up by an area depicting the contours of the displays of all connected computers. Each computer's screenspace is represented by a colored rectangle, sized according to the desktops resolution. Additionally the name of the computer is shown within this rectangle. Using drag and drop, the user can arrange all computers in a two dimensional space. This layout is required for pointer navigation with input redirection, which we will discuss in section 3.2.2 in more detail. Each user can be assigned a different color, which is used, for example, for rendering cursors and visual links.

Furthermore different components of the system are started and configured automatically. This includes in particular input redirection with Synergy (section 3.2.2) and the glass sheet for rendering visualizations on top of the desktop (section 4.3).

4.1.3 Browser Extension

By using browser extensions or plugins, it is possible to extend the functionality of browsers without the need to modify or recompile the browser itself. We created a plugin for the Firefox[1] web browser. It automatically connects to the daemon and provides functionality to get information about opened websites and interact with the browser. Using WebSockets for the connection allows for a biderectional communication with the daemon.

Render website An image of the currently display website can be rendered. By using the HTML 5 Canvas API[2]and the Firefox specific CanvasRenderingContext2D.drawWindow()[3] function, it is possible to not only render the current viewport but the whole loaded website. The resulting image is sent back as binary message to the client that has requested the image.

Get content bounding boxes Clients can request coordinates of the bounding boxes of specific parts of the displayed website. There are three possibilities to specify parts of the website:

- The current mouse selection of the user
- All occurrences of a keyword
- Specific sections addressed by XPaths[4]

Retrieving the bounding boxes of the current mouse selection or the occurrences of a keyword is based on the implementation of the work by Waldner et al. [Wal+10]. To search for a keyword, Document.evaluate()[5] is used to find it within all text nodes of the doucment. To calculate the bounding boxes, Waldner et al. added an element into the Document Object Model [WHA98] (DOM) tree around each occurrence or the current selection and then calculated the bounding boxes from them. Calculating the bounding boxes without modifying the DOM tree was not possible, because the API only allowed to retrieve the bounding box of whole elements, which can potentially contain much more text than just the requested keyword. We instead use the newly available Selection

[1] https://www.mozilla.org/firefox/
[2] https://www.w3.org/TR/2dcontext/
[3] https://developer.mozilla.org/docs/Web/API/CanvasRenderingContext2D/drawWindow
[4] https://www.w3.org/TR/xpath-30/
[5] https://developer.mozilla.org/docs/Web/API/Document/evaluate

API [Niw17], which does not require to modify the DOM tree anymore, but instead allows to calculate bounding boxes of every possible part of the document. It allows specifying ranges by two nodes and offsets in the DOM tree or retrieving it from the current selection. Using Range.getBoundingClientRect()[6] the performance overhead is now minimal and significantly lower than before. Leveraging the new selection API, we can provide the possibility to get the bounding boxes for any given selection specified by two XPaths and offsets. An example of a message specifying a selection inside a document is shown in Listing 4.2.

```
{
    "selections": [
        {
            "start-node": "./ARTICLE[1]/P[2]/text()[1]",
            "start-offset": 277,
            "end-node": "./ARTICLE[1]/P[2]/text()[1]",
            "end-offset": 311,
            "type": "dom-range"
        }
    ],
    "title": "Military Weapons Missing",
    "url": "file:///home/user/userstudy/vast14-mini-challenge-html/
       MC3/02287.html"
}
```

Listing 4.2: Example data of a selection inside a document of one of our user studies. It contains the URL and title of the document and the start and end nodes and offsets of a single selection.

Report changes All changes with an influence on the displayed content are reported to the daemon. This includes scrolling, zooming, loading and reloading pages and changing tabs. These events are detected by registering different event listeners with the browser.

Remote Control Browser On receiving specific messages the plugin can control certain aspects of the browser. It can open new windows or tabs, change the size or position of a browser window, or scroll the contents of a displayed website to a specific position.

[6] https://developer.mozilla.org/en-US/docs/Web/API/Range/getBoundingClientRect

Drag & Drop Common web browsers allow to drag the URL of the currently displayed website to other applications. But information about the URL is not always enough. Additionally, drag and drop does not work reliable with MPX, which we require to support multiple users on a single computer (as we will discuss in detail in section 4.2.1).

The plugin installs a custom drag handler, which is called upon dragging the URL or the current mouse selection. In this handler, we add additional data to the DataTransfer[7] object, which then can be accessed from the window that receives the drop. The data that is sent includes, in addition to the URL or selected text, the position of the current mouse selection (using start and end nodes, as shown in Listing 4.2), the document title and its favicon. With multiple pointers, the data associated with the drag was always the data of the primary pointer. To be able to retrieve the data of the correct drag, the plugin also sends the data to the daemon which knows which pointer initiated the drag.

Context menu, key listener, toolbar icon The plugin can also extend the browser context menu, add toolbar icons and listen to key and mouse events triggered inside the browser, which is used by many parts of our system:

- For adding observations to the OG (section 3.3.1), a context menu is provided for selected content (Figure 3.8). The same menu can also be invoked from a button in the browser toolbar.
- For cloned browser windows (section 3.2.3), additional controls in the browser toolbar (Figure 3.5) indicate the state and can be used to change the synchronization with the clone's source.
- Visual links (section 3.4) can be triggered by pressing CTRL with a browser having input focus or by pressing a button in the browser toolbar.

Identify Window For certain tasks it is required to identify the window of an application. But for safety reasons browser plugins usually do not have any access to the underlying window manager and operating system. Nevertheless we found a way to identify the process ID of a Firefox instance. Using this process ID, the daemon is able to identify the browsers window (see section 4.3.1). Firefox allows plugins to locate certain files[8], which includes the lock file of the instance of the browser the plugin is running within. This

[7] https://www.w3.org/TR/2011/WD-html5-20110113/dnd.html#the-datatransfer-interface
[8] https://developer.mozilla.org/docs/Mozilla/Tech/XPCOM/Using_nsIDirectoryService

lockfile is a symbolic link with a non existing target, but its name includes the process ID of the running browser instance.

4.2 Seamless Collaborative Desktop

In this section, we describe how we implemented support for multiple users on one computer and seamless across desktop boundaries to meet the requirements identified in section 3.2.

4.2.1 Multi-user Input

We use the Ubuntu[9] operating system which uses the X-Window System [SG86] with support for MPX. This allows using multiple sets of input devices on one computer [HT08]. It supports multiple concurrent window foci and does not require any modifications to existing applications. Each window is assigned the focus of one ore more sets of input devices and receives the events of all these devices without a way to distinguish between them. To distinguish between the input from different devices, applications have to specifically handle MPX input event messages. Whenever an application queries data from an input device and multiple devices are present, data from the application window's *client pointer* is returned. The client pointer is either specified explicitly or otherwise chosen randomly from all devices focused in the window. This automatic selection did not match our expected behavior, so we followed a recommendation of the creator of MPX and explicitly set the client pointer each time a pointer interacts with a window to this very same pointer. This allows users to intuitively change the input focus by mouse click, as they are used to from their own computer with single user input.

Cursor Unfortunately, MPX is not widely used and therefore support for it in common window managers is not always reliable. For example, rendering multiple mouse cursors without flickering was not possible. To overcome this problem, we hide the mouse cursors rendered by the operating system with *XFixesHideCursor* from the XFixes[10] extension and render cursors as small Qt Windows. An event handler for MPX mouse move events is globally registered with the operating system and moves these windows to the locations

[9] https://www.ubuntu.com/
[10] https://cgit.freedesktop.org/xorg/proto/fixesproto/plain/fixesproto.txt

of the real pointers. Furthermore it is only possible with the XFixes extension to get the currently assigned cursor image of a pointer. Unfortunately this is only possible for the first pointer [Wal11]. Because of this, we show always the same cursor image in the color of the assigned user. Additional content can be attached to the cursor, for example to show a window preview on dragging (see section 4.2.2).

Clipboard Modern operating systems allow users to transfer data between - and also within - applications by using a clipboard. Typically, the current mouse selection can be copied into the global clipboard by pressing Ctrl + C and inserted later at the new cursor position - possibly in an other application - by pressing Ctrl + V. The X Window System usually offers a second clipboard, always containing the current mouse selection[11]. Even with MPX, there are, unfortunately, only one or two global clipboards available. To prevent users inadvertently interfering with each other when using the same clipboards, we provide a separate pair of clipboards for each user, which we will describe in detail after introducing the open-source software Synergy in the next section.

4.2.2 Input Redirection

We address basic input redirection with the open-source software *Synergy*[12]. Events from the physical input devices are intercepted and injected into the event manager of the destination computer. On the destination computer, the input events appear to be coming from virtual input devices, while the events are actually received over the network.

We use a Synergy version, modified by Dokter [Dok10] to use a peer-to-peer architecture, rather than the standard client-server architecture. The standard version of Synergy installs as a device server on the computer where the physical input devices are attached and device clients on the remote controlled computers. Our modified version installs the same software component on each computer, which combines the capabilities of Synergy server and client: On the one hand, every computer injects events from the attached input devices into the network, tagged with one particular user's identity. On the other hand, every computer receives events from all other computers and determines if they concern the desktop area managed by that computer, based on the spatial model we provide.

[11] https://specifications.freedesktop.org/clipboards-spec/clipboards-latest.txt
[12] http://synergy-project.org/

Additionally to the modifications of Dokter, we improved and added several features of Synergy:

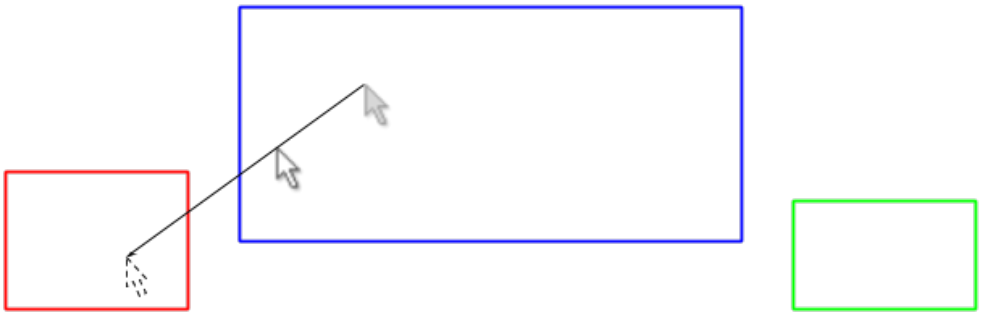


Figure 4.3: The screen space of all computers is mapped into a 2d space. For this example we use the same setup as shown in Figure 3.2. Each computer's screenspace is represented by an accordingly shaped rectangle. The current position of the cursor (inside the blue rectangle) is extrapolated using its position a certain time in the past. Based on the resulting intersection the cursor is warped to the left computer's screen (red rectangle).

Navigation To setup the navigation between the different screens, Synergy is configured to have links between the edges of these screens. For example, the right edge of the screen of the first computer can be linked to the left edge of the screen of the second computer. For more complex arrangements this simple mapping might not be powerful enough. For this, we built a new navigation system into Synergy, where all the screens are arranged as rectangles in a two dimensional space. When leaving the screen, the movement of the pointer is extrapolated as a ray and the intersections of this ray with the other screens rectangles calculated (see Figure 4.3). If there is a hit, the pointer is warped to the closest intersection. This allows for an intuitive navigation between all screens as long as they are organized in such a way, that they can be approximately mapped into a two-dimensional space.

Configuration Synergy had to be configured with a static configuration file before running it. Furthermore, it was designed for controlling multiple computers from a single computer. One Synergy server was launched on the main computer, and, on each auxiliary computer, a Synergy client was started.

For dynamically adding and removing computers from our system this approach was not flexible enough. To provide the required flexibility, we modified Synergy to combine a Synergy server and any number of Synergy clients in a single application. The number

of clients can be changed at any time. Furthermore, we extended Synergy to allow configuration changes sent to a listening TCP socket. JSON encoded messages can be sent to connect to launch additional clients or disconnect any active client, or to update the geometry of any client. The configuration GUI (see section 4.1.2) uses such a socket connection to dynamically configure input redirection. On computers which are set as accessible from other computers, a Synergy client is started. On computers with disabled access, any possibly running Synergy client is stopped.

Input focus As elaborated in section 4.2.1, every window on the desktop has an implicitly or explicitly chosen *client pointer*. Synergy intercepts all input events; therefore, we used this hook to change a windows' client pointer whenever a user clicks into its area on the screen, by using the MPX function call XISetClientPointer[13]. We could use this also to implement access control to applications, and, prevent remote users from changing local system settings; however, for our purposes this was not required.

Multi-user clipboard In the X Window System clipboard operations are implemented using Selections[14]. First, whenever the user request to put content into the clipboard, the according application acquires ownership of the selection, but without copying any data yet. Afterwards, the user can request to insert the contents of the clipboard into any other application. To accomplish that, the application requests the actual contents of the selection from the application currently holding the ownership of the global selection. To enable the transfer of the contents of the clipboard to other computers, Synergy installs a global event listener with the operating system to receive any clipboard related events. Whenever an application acquires ownership of the clipboard, Synergy immediately requests the contents of the clipboard and afterwards acquires the ownership of the clipboard on its own. If the user then changes to a different computer, Synergy transfers also the contents of the clipboard to that computer and provides them to any application requesting data from the clipboard.

As this mechanism is still limited to one global clipboard we extended it to support an arbitrary number of users on the same computer as well as on multiple computers. For this purpose, we extend the Synergy protocol to include the pointer id for each message related to clipboard synchronization. Unfortunately, the operating system is not able to differentiate between different sources, even when there are multiple pointers active.

[13] https://linux.die.net/man/3/xisetclientpointer
[14] https://tronche.com/gui/x/icccm/sec-2.html

To overcome this limitation whenever someone copies something into the clipboard, we use XIGetClientPointer()[15] to get the client pointer of the window which currently holds the clipboard. Then we copy the clipboard to an internal storage tagged with the according user's id. Immediately after reading the contents of the clipboard, Synergy takes ownership of the clipboard, such that a later request to paste the content of the clipboard will be handled by Synergy. When the contents of the clipboard are requested, we check which user's cursor is active in the requesting window and sent back the user's contents previously copied into the internal storage. When a user navigates with the cursor from one computer to another, the internal storage of the user's clipboard is transferred to the new computer, providing a seamless clipboard across all participating computers.

One issue with this approach is that it will not properly work when multiple users interact with the clipboard at exactly the same time. But as the usual timeframe required to interact with the clipboard is only a few milliseconds, this is normally not an issue. To completely ensure interference can not happen, modifications to the operating system would be required.

Window dragging across computer boundaries Drag and drop with the X Window System is implemented using the XDND protocol[16]. Unfortunately, available software often does not fully comply with the specification. For example, the Firefox web browser did not accept drop events when they where not sent as result of hardware event. Events injected by virtual input devices where simply ignored. For this reason, we do not support general drag and drop across computer boundaries, but instead provide the ability to move browser windows across computer boundaries. Clicking on a window while holding Ctrl initiates window cloning. The browser plugin sends a message with information about the currently displayed content (Listing 4.3) to the server, which in turn grabs the mouse to be able to catch the next mouse click anywhere on desktop. While the mouse is moving, a preview image of the dragged window is attached to the cursor. When a cursor moves to a different computer, the drag data and its preview image are transfered to the other computer as well, to keep the illusion of a seamless dragging operation. With a mouse click the dropping process is initated. A new browser instance or tab is opened, showing the same URL and location. The (unique) ID of the

[15] https://linux.die.net/man/3/xigetclientpointer
[16] https://freedesktop.org/wiki/Specifications/XDND/

source browser window is also transferred, so that synchronization (as we discuss in section 3.2.3) is possible.

```
{
    "task": "drag",
    "url": "https://docs.google.com/document/<document-identifier>",
    "scroll": [0,0],
    "elements-scroll": {
        "./DIV[4]/DIV[0]/DIV[0]/DIV[7]/DIV[0]": [0,884]
    },
    "view": [922,105,572,426],
    "tab-id": "55dc2c78fcfa42cb00e54137325ef91939d2ceb5",
    "type": "browser",
    "screenPos": [1021,418]
}
```

Listing 4.3: Example data of a message initiating a document clone operation.

MPX was originally intended for use with multiple physical input devices connected to the same computer, and Synergy was designed to control multiple computer with a single set of input devices. By combining both technologies, we arrive at the combined effect of being able to control any application on any display connected to any computer using any input device. Per default, we make all displays accessible. However, if a user desires privacy for a personal display, such as a notebook computer, the users allowed to enter the personal display can be restricted.

Of course, using multiple input devices in the same application instance only works with applications designed for multi-user operation, which is rarely the case. Fortunately, this restriction turns out to be of little practical consequence. Concurrent operation inside one application instance is rarely an important requirement for analysis work. Collaborators mostly take turns between working in parallel with separate documents and discussing common findings, with one user being active at a time [GGoo]. With ample display space, joint inspection and manipulation of the same document can be carried out in two synchronized windows, as we have explained in section 3.2.3.

4.3 Glass Sheet Visualizations and Annotations on the Desktop

4.3.1 Identify and Monitor Windows

For annotating content on the screen, it is a basic requirement to know the contents position on the screen. When extending applications by plugins (as in section 4.1.3), for security reasons, access outside the applications window is strongly restricted. It is often even not possible to retrieve the position of any window content in absolute screen coordinates, but only in coordinates relative to the application's window. Solving this problem requires to find the applications window geometry. Our server has access to a list of all open windows including their unique ID, process ID of the application, title and geometry. If the operating system's window ID is accessible to the plugin API, then identifying the correct window is trivial. But most of the time, another way to identify the correct window is required. For this purpose, we first retrieve a list of all open application windows and then filter step by step by the following criteria:

- Process ID (sometimes it is possible to retrieve the process ID of the running application. See section 4.1.3)
- Window geometry (ignore title bar of maximized windows, because they are integrated into the global menu bar)
- Window title (User programs and window managers often add their name or other information to the title, so we only check if the window title starts with the title as reported by the plugin)

As soon as only one match is left, the filtering process is stopped. If more then one window is left, the top-most window is selected. This usually works, as the top most window is most likely the window with which the user has interacted with most recently. When more than one matching window remains, the filtering process is repeated every time the server detects a change in any of the opened windows. Once a unique match is found, the process is not repeated again.

Knowing the windows geometry allows to render annotations on the screen and to detect window overlap, which is needed by our hidden content visualization techniques, described in section 3.5.

4.3.2 Retrieve Geometry of Applications Taskbar Icon

Rendering annotations next to the icon of an application in the taskbar, for example to indicate hidden content in minimized applications (Figure 3.16a), requires knowledge about the icon's geometry. Unfortunately, this information is not exposed by the window manager, and a thorough search for existing solutions yielded no results. For example, on Ubuntu, the operating system we are using, the BAMF[17] daemon is responsible for selecting the icons for running applications, and afterwards the Unity[18] launcher places them in a side panel. However, apart from the list of running applications, no further information about order or exact placement of icons on the screen is exposed.

To work around this limitation, we investigated how icons are placed within the launcher (Figure 4.4) of the Unity desktop environment. The size of the launcher can be configured by the user; however, it is possible to retrieve the geometry of the window displaying the launcher. This is achieved by identifying that window through its unique title, using the method described in the previous section 4.3.1. From the size of the launcher window, we derive the size of the individual icons and the geometry of the area where icons are displayed.

On top, a list of the user's manually pinned or locked favorite applications is shown, followed below by icons of all other currently running applications. Ubuntu uses dconf[19], a key-based configuration system. Querying it for the key */com/canonical/unity/launcher/favorites* returns the list of all favorite applications. After removing all non-alphanumeric characters from an applications name, it can be matched against the entries in that list. The previously calculated size of the icons and the position of an application in the list leads to the actual position of that applications icon in the launcher.

4.3.3 Visual Links and Hidden Content

These other applications are integrated using a minimally invasive approach through the use of a plug-in API or minor modifications of the application source code, if no API is provided [Wal+10]. The data exchange between the server and the clients is handled using WebSockets. To add the described user interface components on top of the existing screen content, we create a transparent Qt window which we render using OpenGL.

[17] https://launchpad.net/bamf
[18] https://launchpad.net/unity
[19] https://wiki.gnome.org/Projects/dconf

Figure 4.4: Ubuntu Unity application launcher. The geometry of the launcher window (red rectangle) can be retrieved from the operating system. The width is user configurable and the padding is fixed, therefore we use the width to calculate the height of the icons. With this information and the list of currently running applications we can calculate each individual application's launcher icon's geometry.

After the server has received a user-triggered search string, it forwards the request to every connected client application, enabling each application to add its regions. Upon receiving a request, each client searches its content for instances of the requested identifier and reports back the bounding boxes of all found occurrences. For simple selection types, like individual words, bounding boxes already provide an accurate approximation of the relevant region. To highlight and link more complex shapes, such as objects in a map (as can be seen in Figure 3.12) or graph, a client is free to use arbitrary polygons for representing its regions.

Visual links are drawn between all highlighted regions. The naive approach that connects all highlighted regions to a common center results in a cluttered visualization. To remedy this, we bundle links using force-directed edge bundling [HW09], an algorithm based on an iteratively refined system of control points, attracting each other. The system is initialized by calculating the center of gravity of all occurrences. Then, the highlight region closest to the center of gravity is determined, and all other regions are connected to it. Moving the center of gravity avoids an artificial branching point. Next, all links are subdivided into segments of approximately equal length and finally, force-directed edge bundling is applied. Due to potentially large differences in the length of individual links, the forces affecting a single link can change rapidly, leading to sharp corners in the link routes. To address this issue, we apply a geometric smoothing on the points forming the

link routes after executing the bundling algorithm.

All rendering output is directed to an off-screen buffer first. This buffer is four times the size of the screen and copied into the fullscreen window. Using hardware accelerated texture filtering, the visualization is automatically smoothed while being downscaled. If the hardware and operating system allow proper blending of OpenGL output directly with the desktop, then it is possible to skip the additional rendering step into the offscreen buffer. This greatly reduces memory usage, but, of course, also creates less smooth visualizations.

The use of alpha-transparency allows blending with the desktop content. Screen pixels that are not covered by the visualization are masked, to allow mouse events passing through. In this way, the user can interact with all content that is not covered by our visualization.

For rendering see-through visualizations and smart previews, the client application is required to send an image of its content to the server. We use a hierarchical tile map, where each level consists of a single or multiple tiles, which add up to a full preview image of the client application's content at a specific zoom level. Using different resolution images for the individual zoom levels, we create tiles in a resolution that is sufficient for a single level of zoom. As the user zooms into the preview or moves the viewport, missing tiles are asynchronously requested from the corresponding client application.

While working in a desktop environment, the arrangement of opened windows can change at any time. As this possibly affects the position and occlusion of regions, we need to react to such changes. Current operating systems usually do not allow receiving notifications for changes in windows of other applications. As a workaround, we use a window monitoring component, which periodically requests a list of all opened windows, including their geometry and stacking order, and compares this information with the previous state. If any changes are detected, the server triggers the recreation of the visualization for all active identifiers.

As a proof of concept, we have integrated several widely used applications into our system. A browser add-on allows searching for words or phrases matching a given search identifier. The bounding boxes of all found occurrences, both within and out of the current viewport, are sent back to the server for further processing. As a non-textual

example, we have used the Google Maps JavaScript API[20] to create a mash-up which supports the search for geographic locations by name and the retrieval of corresponding screen coordinates on the map. Retrieving the name of a location on a map and querying by this name is also supported. (Unfortunately it is currently not possible to retrieve the outline geometry of search results. Data could be queried from OpenStreetMaps).

For connecting minimized windows and the desktop search engine, the location of the associated icons needs to be known. Therefore, we query the list of windows in the task bar and calculate the exact location of each icon using the known icon sizes (section 4.3.2). To retrieve the data for the smart preview for unopened files, they are opened in the background in their respective applications, while visual feedback is suppressed.

For a basic integration with our system (to show the number and location of elements) applications need to implement a simple WebSockets protocol, which all modern browsers support. For a full integration, applications need to provide imagery of hidden areas for the preview, which is typically supported in either the GUI library or the graphics API.

4.4 Observation Graph and Deep Links

The OG viewer is implemented as a web-application, using HTML5 and the D3 [BOH11] library for rendering. It runs in a web-browser and communicates with other web-browsers (and the documents or web-based applications therein) through a plug-in for a standard web-browser (Mozilla Firefox), using JSON and WebSockets. The main state of the OG (observations with metadata and links) is fully synchronized, while every OG instance can have additional individual state per user and per browser window. Other web-based applications can send messages to modify the OG, and all opened OG instances in the cluster get notifications. The resulting approach is minimally invasive: Installing the plug-in, which only uses the standard API of the browser, is sufficient to integrate any web application with the OG system. Existing web applications do not require any modification and continue to work normally.

References to specific selections in a document are stored in a record consisting of the document's URL and two XPath pointers (plus character offset) bounding a section of the

[20] https://developers.google.com/maps/documentation/javascript/tutorial

DOM. This approach works for static web pages and many dynamic web applications, as long as the DOM does not change in a way that invalidates the XPath pointers.

Our minimally invasive web-based approach differs from previous document analysis software, which imports all document sources into a monolithic software framework, replacing standard desktop operations, such as window management or scrolling in the browser, with proprietary ones. Our approach works for all common DOM elements, including SVG-based visualizations. For example, we conceptually could easily store and re-identify a selected bar in a bar chart or other visual glyphs.

The communication between the different browser windows is implemented in two variants: Either the web-browser instances communicate with each other in a peer-to-peer manner or via a service process. The peer-to-peer variant makes our implementation fully independent of the underlying operating system, since only a web-browser is required. We favor this variant, if portability and easy deployment are most important.

The variant using the service process requires an initial software installation, but provides three important additional capabilities: First, it optimizes placement of windows containing links to the graph, so that they are close to their referring node.

Second, it draws visual links between a node shown in the OG viewer and its source section on the glass-sheet. We include visualization techniques for hidden content [Gey+14], in which visual links connecting to relevant entities are rendered, and pointers to occluded occurrences of a search term are added. OG and the visual links can be employed independently of each other.

Third, the service process synchronizes OG among multiple users occupying separate computers in a local network.

5 Experiments and Results

In this chapter, first, we report on explorations and evaluations of how individual users perform sensemaking tasks when they are given a choice of implicit and explicit structuring of information. Second we report about informally collected feedback from a pair of users on the same tools and task as in the first evaluation. Finally, in the last section, we report about two user studies with the goal to compare the effectiveness and efficiency of our hidden content visualization techniques (as described in section 3.5) and traditional search for finding hidden content. The first study had a low, and the second study, a higher number of simultaneously opened windows on the desktop.

5.1 Observation Graph

In this section we report on two exploratory evaluations of our system. Individual user were asked to work on an intelligence analysis task on a large display. We observed the influence of using an observation graph (OG), as described in section 3.3, in comparison to conventional note taking in a Google Docs document[1] on display space usage [Gey+17]. We assumed the graph itself would allow users to externalize their mental models and use spatial encoding in a more compact manner than by spatially arranging entire application windows. In addition, we aimed to explore the different sensemaking strategies by the users through observations and subjective feedback.

5.1.1 Study Design

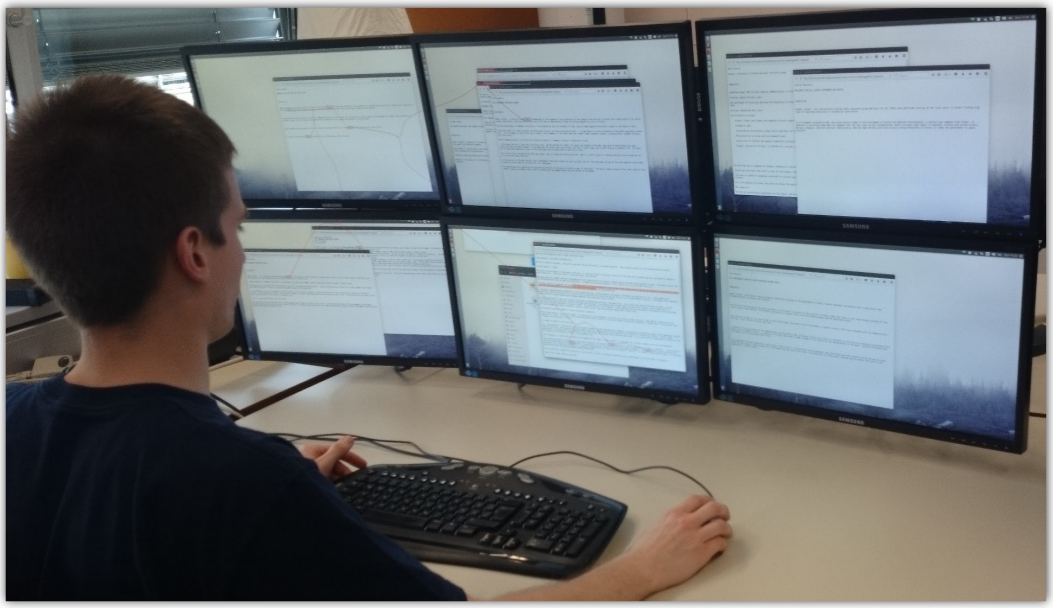


Figure 5.1: A desktop on a tiled display. Six monitors connected to a single desktop PC serve as a large display in an office.

The studies were conducted on a standard desktop computer (Intel Core i7-930 CPU @ 2.8GHz, AMD Radeon HD 7870 GPU, Ubuntu, Firefox) connected with six 22-inch monitors with a resolution of 1920x1080 pixels each. The monitors were placed a regular office desk in a 3×2 arrangement. The user was sitting 70cm away from the central monitor. The display setup is shown in Figure 5.1 and was 155cm wide, covering a visual angle of 95°.

[1] https://docs.google.com

Conditions

In a between-subjects design, we assigned each half of the users to one of two conditions:

In the **control condition (CC)** documents were shown in regular browser windows. To search through the data, we provided users with Recoll[2], a browser-based full-text search tool. Selecting a document in Recoll opened it in a new window with the same size as the Recoll window. In addition, visual links, as described in section 3.4, could be invoked to connect text fragments between all open document windows. Links could be triggered from a selected word, phrase, or by searching for a term. In contrast to Andrews et al. [AN12], we did not perform named entity extraction for visual links, to avoid introducing a confounding factor between the study conditions. We chose to use visual links, as they have been shown to improve performance when recognizing related items on the screen compared to simple highlighting [Ste+11] — an aspect that is especially pronounced in large display setups, where conventional highlighting might be missed if located outside the user's field of view. Users were provided with an empty Google Docs document to take notes.

In the **OG condition (OG)**, users could record information by creating nodes and edges in the graph and by adding notes to them. Linking between the OG and the document windows, with references to exact sections in the document, allowed for quick switching between the graph and the source information. Visual links could also be used for connecting arbitrary selected text between documents, as in CC, but also to connect node labels in the OG with the web documents. No separate text document for note-taking was provided.

OG differs from CC in the following aspects: (1) it provides *visual abstraction* of the contained information, (2) it organizes the information in a *graph structure*, (3) it provides *linking* between the graph nodes and edges and their associated source information, and (4) it *automatically places document windows* according to the user's graph layout.

We chose a between-subjects design, as this allowed us to use only one task, limit the length of the analysis session, and avoid learning effects. On the downside, between-subjects designs can distort the results due to individual variability. We will therefore not only report statistical significances, but also present the quantitative results visually, and provide qualitative results.

[2] http://www.recoll.org/

Procedure

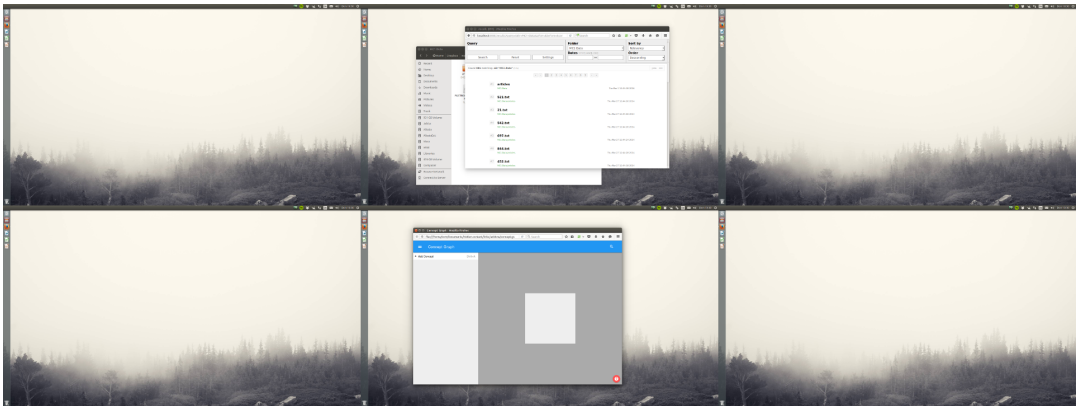


Figure 5.2: Initial setup of the desktop for the OG condition. The window on the center-bottom monitor contains an empty OG. The center-top monitor displays a file browser and the desktop search engine. The initial setup for the control condition was identical, with the exception that OG was replaced by a Google Docs document for note-taking.

At the beginning of the session, as shown in Figure 5.2, the Recoll window was placed in the middle of the upper central monitor. For OG, the empty OG window was placed on the lower central monitor. For CC, an empty Google Docs document was used instead.

Users first were introduced to the tools using an unrelated data set. The search tool and the use of visual links was introduced for both conditions; the OG was introduced only for participants in the OG condition. After the introduction, users were asked to replay the demonstrated actions and encouraged to ask questions about the setup. They were free to test the system as long as they needed to get familiar with it.

The subsequent analysis session was limited to an hour, after which users were asked to present their intermediate results. After they completed the session, we asked users to fill out a questionnaire, followed by a semi-structured interview. Moreover, sessions were video-recorded, and all graph activities (concept or edge creation, adding or removing references), link activities (creation and deletion), window activities (opening, closing, moving, resizing), and keyword searches were logged.

5.1.2 Pilot study

This study was conducted with 10 users. All users were recruited from a local university and had a background in computer science. Nine users were male, one female, aged 26 to 40. Two users normally work on a 15-inch laptop, two users use a single 22-inch

monitor, and all other users typically work on two monitors ranging from 22 to 27 inch. As computer scientists, users were familiar with sensemaking tasks, such as related work research. All users reported to use one or more dedicated tools for such tasks: Five users frequently use dedicated note-taking or reference management tools. Five users use general-purpose tools like text files or sheets of paper instead of, or additionally to, note-taking software, and five users reported to use browser tabs or multiple application windows to organize information sources for sensemaking.

We used the task descriptions and data from the VAST 2014 MiniChallenge 1 [CGW14]. The data comprised over 800 text articles, as well as some resumes, a map, an organization chart, and two spreadsheet documents. We used the first question from the MiniChallenge as the analysis task, which asks participants to identify leaders and important members (actors) of a potential terrorist network called POK.

Results

To analyze the influence of the sensemaking tools on the analysis task and on display space usage, we first discuss the observed performance differences, then report observations and feedback on the usage of the sensemaking tools, and finally report differences in display space usage.

We compared the questionnaire items, the usage frequencies of the tools, as well as the number of correctly identified members in the analysis task by Independent Samples t-tests. The display usage parameters (average and maximum number of open windows and display coverage, respectively) were compared using a MANOVA. We report significant differences, but do not mention explicitly whenever differences are not statistically significant. Due to the low statistical power of the experiment caused by the small sample size, we cannot reject the null hypotheses for non-significant results with a high certainty. We therefore also report the mean differences along with a visual representation of the results.

Task performance To assess the task performance of the users, we counted the number of correctly identified members of the potential terrorist network. On average, OG-users and CC-users scored similarly, with a mean of 6.0 and 6.8 correctly identified members, respectively (see Figure 5.4). Users' subjective satisfaction with the outcome was also

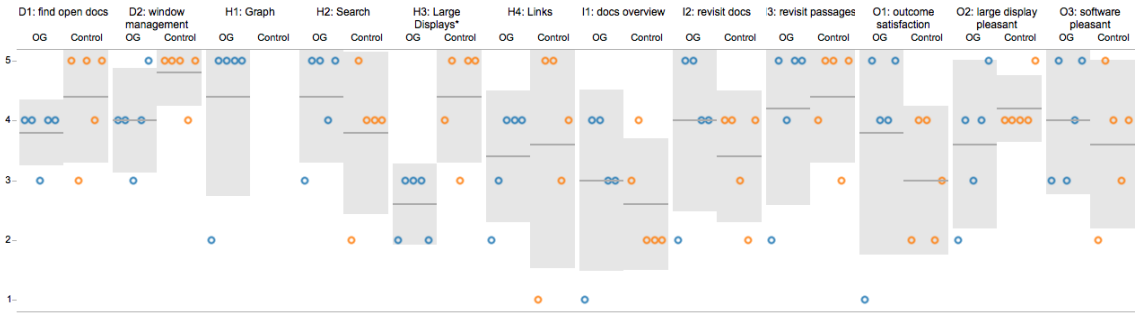


Figure 5.3: Comparison of survey responses for the two conditions. Circles represent individual answers, the horizontal lines represents the average for each condition, and the gray boxes represent 95% confidence intervals. Answers were given on a 5-point Likert scale ranging from 1 to 5. High values indicate a positive rating (e.g., helpful, easy, good, pleasant) and vice versa. There was a significant difference in responses to H3 regarding the helpfulness of the large display, which was rated higher in the control condition. The graph (only asked in the OG condition) was rated very helpful except for by one participant.

rated similarly, with 3.8 for OG-users and 3.0 for CC-users on average, on a 5-point Likert scale (see Figure 5.3 O1).

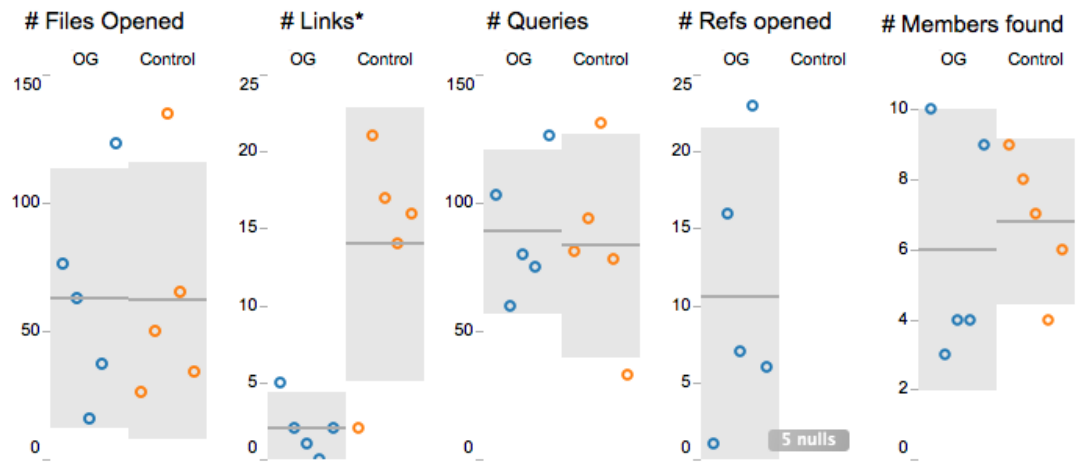


Figure 5.4: Number of files opened, number of times links were used (significant difference), number of search queries executed, number of references used out of OG, and number of actors found (the task, out of 20 actors total) for the OG and control condition.

Sensemaking tools usage Users opened a similar number of files (63 for OG-users and 62 for CC-users on average) and conducted a similar number of queries (89 for OG-users and 83 for CC-users on average), as shown in Figure 5.4. They also rated the search tool similarly helpful (4.4 for OG-users and 3.8 for CC-users, as illustrated in Figure 5.3, item H2).

Keyword linking CC-users employed keyword linking significantly more frequently than OG-users (2 vs. 14, on average), even though OG-users were also instructed how to link keywords across document windows and how to link concept keywords from the graph to documents. Usage of visual links also varied strongly between CC-users. CC5, for instance, used the links only twice, and also rated the helpfulness of links much lower than the other participants (see single low value for H4, right column, in Figure 5.3). Instead, he intensively used the Google Docs document and also wrote down the file names where he found the information. He then subsequently worked on various hints and took notes whenever necessary:

> *"Actually, there was always one term I was currently working on and, therefore, I found further things, which I searched for as well."*
>
> (User CC5)

User CC2, in contrast, used links frequently and reported:

> *"When I switched on the links, I immediately found them [the windows] again."*
>
> (User CC2)

User CC1 also appreciated links primarily to find documents again:

> *"In principle, I tried to arrange windows approximately according to the topics. But somehow, that did not work so well. [...] So I tried to find a window again, which I had lost, with the links."*
>
> (User CC1)

By analyzing log files and reviewing video recordings, we found that users also used links right after opening a document to search for the term they had previously used in Recoll to find the document. This would not only highlight occurrences of the search term in this newly created document, but also render links to the other open documents. Afterwards, they read through the new document; if they found some (new) aspects, they occasionally investigated possible relations to other open documents using the links.

Four of the five CC-users utilized the Google Docs document to take notes. User CC1 took notes on paper instead. The participants came up with different strategies how to structure their findings in the document. Users CC2 and CC4 noted a list of names together with a few associated notes. User CC3 tried to reconstruct the chronological order of events on January 20, while user CC5 copied entire paragraphs from the source documents into the file, together with the source file name.

Observation graph From the OG, OG-users revisited documents (i.e., reopening or revealing windows) by clicking on linked references 11 times on average (see "# Refs opened" in Figure 5.4). Like for keyword linking, we observed a strong between-subject variation in terms of usage. While PG1 and PG3 used cross-referencing frequently, PG5 used it only once. User PG5 also issued the lowest scores for questionnaire items I1 (*"I had a very good overview of the documents I had already visited"*), I2 (*"It was very easy to find and revisit key documents again"*), and I3 (*"It was very easy to find the relevant passages in the key documents again."*), as illustrated by the single low values for I1, I2, and I3, left column, in Figure 5.3. This user invested the majority of the study time reading the documents and hardly used OG or the large display space at all. Interestingly, though, he identified 10 network members, which is more than all the other users. However, apart form this user, all OG-users issued the highest possible score for the question whether the graph was helpful to solve the task (Figure 5.3, H1).

Similarly to the different strategies for organizing findings in the Google Docs document by CC-users, OG-users used various approaches to organize their graph. Participants OG1, OG2 and OG4 used nodes for people, organizations, places, events and diseases, with zero to one document reference per node and edges for relations between the nodes, as shown in Figure 5.5.

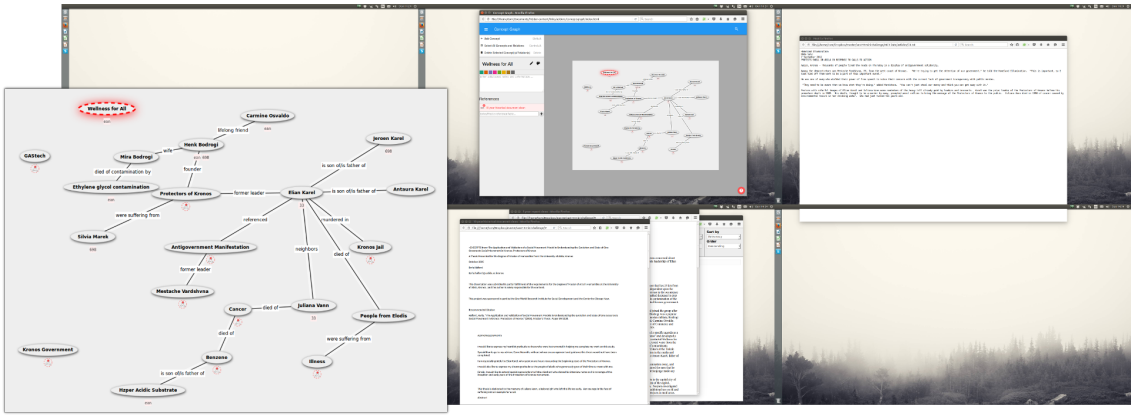


Figure 5.5: Final desktop state of user OG2: only three screens are used. The inset shows the OG in detail (The desktop behind the OG was empty).

The other two users had smaller graphs. User OG3 created two nodes for the two most relevant organizations, connected by the labeled edge "kidnapping". Both nodes were connected to a few nodes describing general concepts like "history", where document references were attached to.

Finally, user OG5 had only four isolated nodes in his graph at the end of the study

("POK", which is the potential terrorist network, "POK leaders", "KIDNAPPING", and "GAStech", which is the affected company), where a few references were attached.

On average, users had 10 nodes in their graph, 10.25 references from nodes to document passages, with 7.75 edges and 7 references from edges. Three OG-users mentioned in the interview that they were still in the process of shaping the graph at the end of the study, so going back to files was rarely necessary. Still, users appreciated the possibility. User OG5 answered the interview questions on whether finding information sources again was easy as follows:

> *"Yes. Once it [the document] was linked [to the graph], one could open it again with once click."*
>
> (User OG5)

Display usage To quantitatively compare display usage between the two groups, we measured the display usage as the percentage of display space covered by application windows in one-minute intervals. The average display space covered by application windows over the entire task of CC-users was significantly higher than of OG-users (51% vs. 23%, see Figure 5.6). This difference was even more pronounced for the maximum display usage with 66% for CC-users and 27% for OG-users, which was also statistically significant.

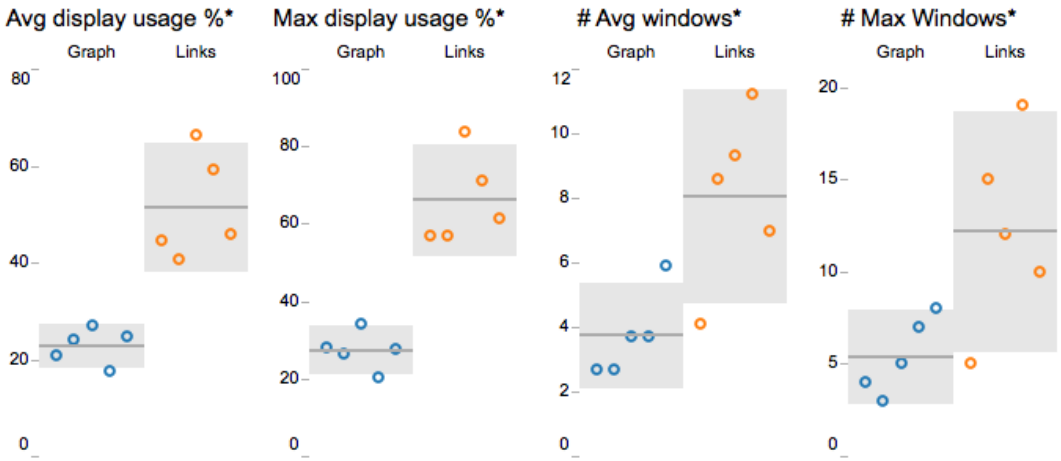


Figure 5.6: Usage of display space and windows in the two conditions. All results show significant differences between the two conditions. Display usage and the number of windows were consistently higher in the control condition.

Figure 5.7 shows heat maps of how the displays were used during the analysis session. The left heatmap shows the OG condition and the right heatmap shows the Links

Condition. The position and size of each window were recorded in one-minute intervals, aggregated for all users. Note how, in OG, the windows are concentrated on the two center screens, while the two screens on the left are never used, and the two right monitors were used only occasionally. In contrast, all CC-users utilized at least five of the six monitors.

In line with our results about display usage, we also found a considerable difference between the number of open application windows in the two conditions. The average number of open application windows was significantly higher for CC-users (8.04) than for OG-users (3.74); see Figure 5.6. The maximum number of open windows was also twice as high for CC-users, which is also a significant difference (5.4 vs. 12.2). Note, however, that the number of opened files (and therefore also the number of opened windows) was almost identical between those two groups, as shown in Figure 5.4. This implies that the difference was not caused by the number of visited documents, but by the way users managed their document windows. We observed that OG-users closed document windows right after reading, and only kept them open occasionally. User OG3 explained this behavior as follows:

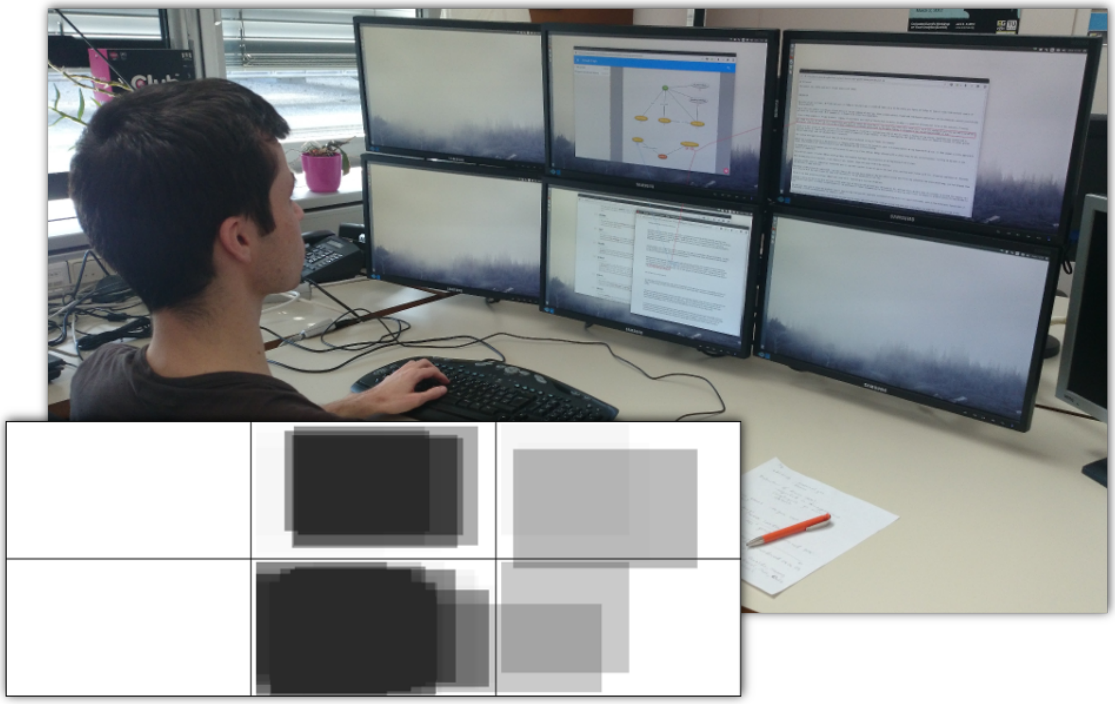
> *"If I had the link [reference in the graph], then I knew, now I only need to look for the [POK] leaders in the graph, and therefore, I close it [the window]."*
>
> (User OG3)

OG-user OG4 stated that she did not need the space to solve the task, and OG5 suspected that he did not use the display space because he is used to less space as he uses only a single 15-inch laptop monitor. Similarly, OG3 explained:

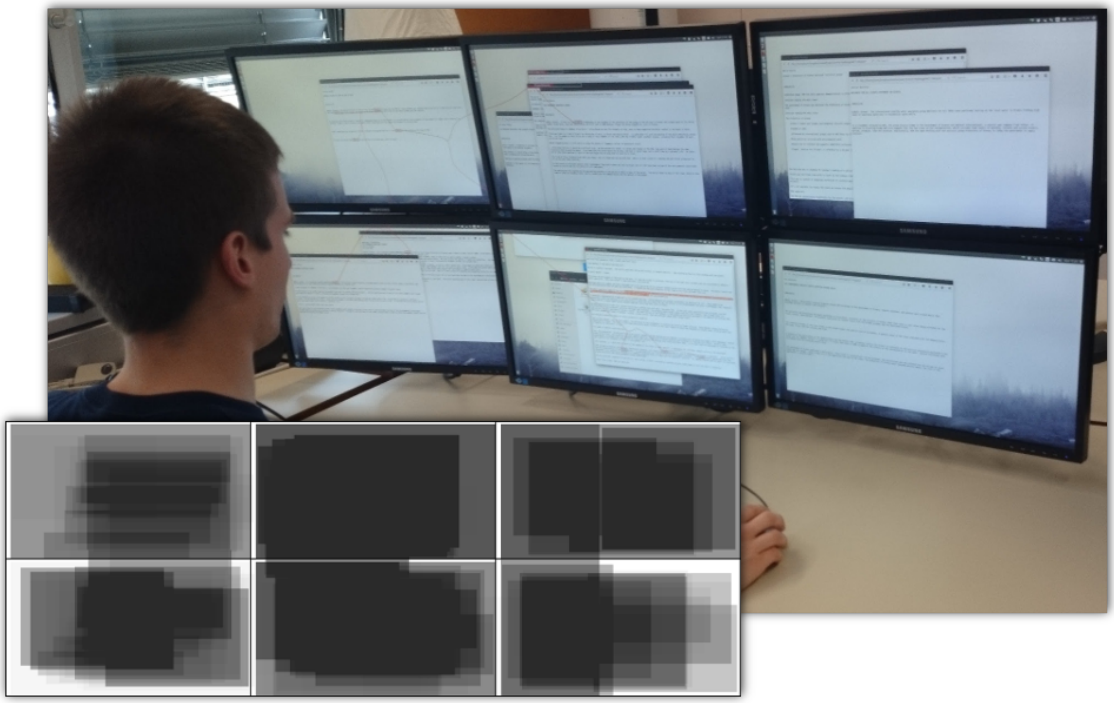
> *"I think I did not use it because I am not used to it, because I only have two [monitors]. I am used to having everything on top of each other."*
>
> (User OG3)

Figure 5.5 shows a screenshot of user OG2 at the end of the study. He never used more than three monitors at the same time. In the interview, he explained that two windows and two monitors were enough: one for the graph, and one for browsing the information.

CC-users tended to keep documents with relevant content open for a longer time compared to OG-users. For instance, CC2 stated:



(a) OG condition



(b) Control condition

Figure 5.7: Heatmap of the display usage in the OG and control condition showing the position and size of each window as recorded in one-minute intervals, aggregated for all users. Note that in the OG condition the windows are concentrated on the two center screens, while the two screens on the left are never used. Windows are much more distributed in the control condition.

> *"I left windows open whenever I thought that the information within could still be somehow interesting."*
>
> (User CC2)

We observed that CC-users exploited the large display space to spatially organize their document windows. This is a well-known strategy of knowledge workers on large displays [BB09; AEN10; Wal+11a]. In the interview, three out of five CC-users could explicitly describe their strategy for utilizing the display space. For instance, CC1 structured his documents mainly around persons. Figure 5.8 shows a screenshot of his desktop at the end of the experiment. CC3 tried to build a timeline and sub-divided the six monitors into three logical groups (current information, search document, and long-time storage). CC4 used the two central monitors as "working screens", and the side monitors as "info screens".

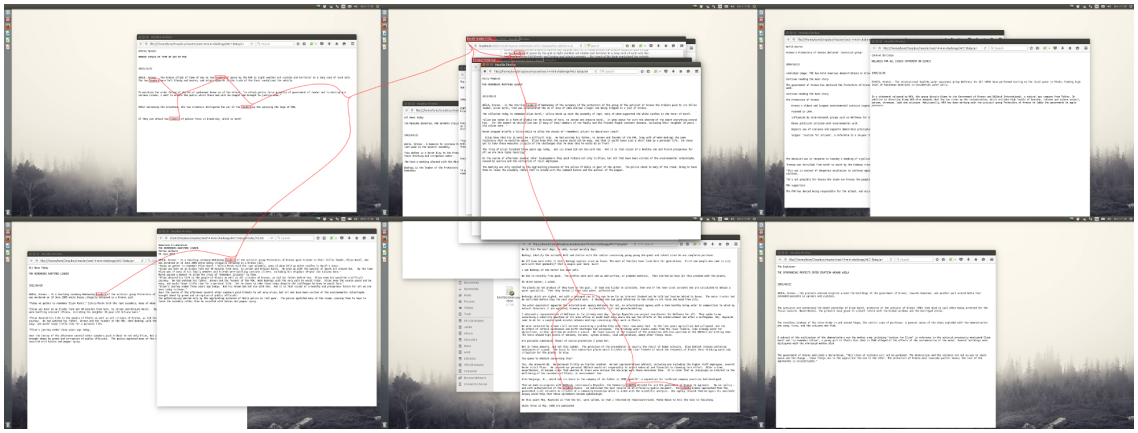


Figure 5.8: Final desktop state of user CC1 with window groups across the entire desktop and active keyword links to highlight related information across open document windows.

These observations are also reflected in the questionnaire results. We found a significant difference between the ratings for the helpfulness of the large display to solve the task between OG-users (average rating 2.6) and CC-users (average rating 4.4, see H3 in Figure 5.3). On average, CC-users also issued higher ratings for the questionnaire items D1 (*"It was very easy to find open documents on the large display"*) and D2 (*"It was very easy to manage (place, re-size...) multiple windows on the large display"*), as visualized in Figure 5.3.

Discussion

It was interesting that users performed equally well in the analysis task, irrespective of the tools they used (OG or Google Docs document). They explored approximately the same amount of documents and issued a similar number of keyword searches. Without OG, a significantly higher number of document windows was opened. Consequently, with OG only a small fraction of the available display space was used. Users without OG all utilized the entire display space to spatially organize document windows.

In terms of performance, we could not observe differences. It seems that the *outcome* of the task was not influenced by the provided sensemaking tools, but the *process* was. Without OG, users exhibited well-known spatial organization strategies with a large amount of document windows on large displays.

Based on these findings, we conclude that spatial organization indeed is a successful strategy to externalize mental concepts. However, we argue that it is not necessary to use wall-sized displays to perform this spatial organization. Instead, a light-weight general-purpose sensemaking tool, like the OG viewer employed in our study, can provide this spatial organization support in an abstracted, spatially compressed format. While spatial window organization is inspired by piling of physical paper on the desk [Mal83], it seems that even a simple tool like OG can provide organization facilities that go beyond clustering and piling. The answer to our initial research question (*"Do users exploit a large display space when using a simple, well-designed sensemaking tool to the same extent as without such a tool?"*) therefore is: *"No, with an appropriate sensemaking tool, users do not (need to) organize their documents spatially on a large display."*

We observed that in the graph condition, users made use of multiple windows and multiple screens mainly for comparisons (i.e., the "work zone") instead of spatial organization and closed documents after they extracted the relevant knowledge. We believe that the ability to return to the relevant passage within the original document was a major factor why users in our study reduced the number of open document windows and, as a consequence, the used display space. As user OG3 explained:

> *"If I had to look something up again (what did I read here?), the graph helped a lot, because it was already organized, and I just click the reference to arrive exactly at the document and look it up again."*
>
> (User OG3)

It has to be noted, though, that maintaining these references is not always technically possible with our current implementation, especially, when a web application makes frequent changes to the DOM structure. To improve the technical ability for retrieving references, we could use different strategies, including caching the document. In cases where referencing or caching is not possible, we expect that users would return to their conventional organization strategies and leave windows or browser tabs open until the end of the sensemaking session.

A question that remains open is how display space usage evolves over time. Our study was limited to an hour, and there was an extensive number of documents to be explored, so users rarely revisited documents. They rather skimmed one document after the other without later verification. Reopening and comparing documents may increase the need for display space to enable effective side-by-side comparison. A longer-term evaluation of display space usage in combination with OG is therefore important future work.

In a longer-term field study one could also explore the limitations of OG. The VAST mini-challenge task we used in this study had an inherent graph structure that was supported well by OG. Due to time constraints, users were only asked to answer the first question concerning the network of people. The second question was about the temporal sequence of events. We expect that OG would provide limited help for this task. However, a similar tool for linear (chronological) organization of concepts and findings could make OG suitable for temporal tasks as well.

5.1.3 Hypotheses

Based on the outcome of the pilot study we formulated the research questions for a second, larger study [Gey+17]. Our hypotheses were that (1) both groups would use spatial organization to externalize their knowledge and that (2) — despite having the same window management — users of the control condition would spatially organize document windows on the large display (as observed by Andrews et al.), while OG users would rather spatially organize nodes of the concept graph.

All 20 users (10 female; age 22-49) were knowledge workers (students, researchers, or administrators). Sixteen users had a computer science background; all users were familiar with sensemaking tasks, such as literature research. Some users reported to have experience with dedicated tools for sensemaking or information management, such as Evernote, Mendeley, OneNote, or Trello.

We used the task descriptions and data from the 2011 VAST MiniChallenge 3 [Gri+11]. The data comprised around 4.500 text articles, of which 13 contained manually generated news regarding a terrorism threat in the fictitious Vastopolis area. The task was to identify any imminent terrorist threats in the Vastopolis metropolitan area and to provide detailed information on the threat.

5.1.4 Results

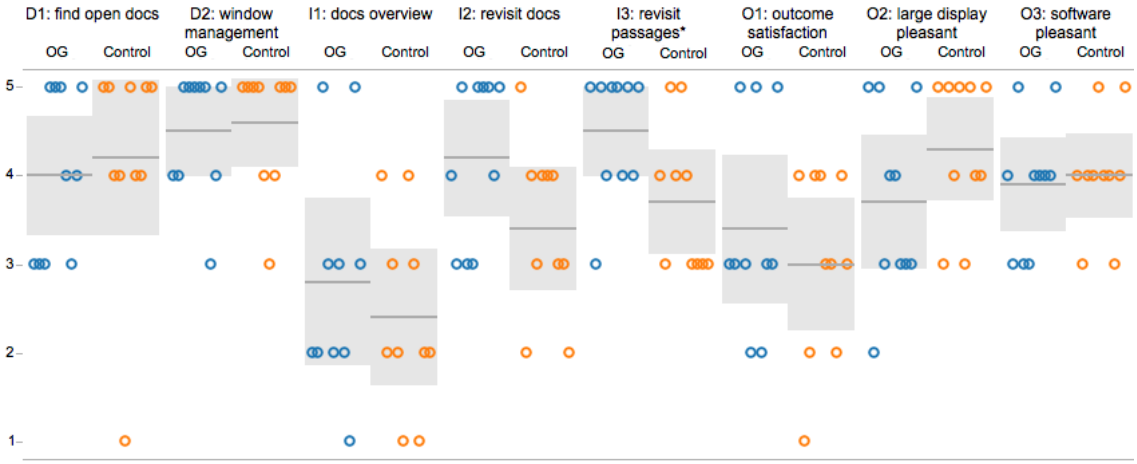


Figure 5.9: Comparison of survey responses for the two conditions. Circles represent individual answers, the horizontal lines represents the average for each condition, and the gray boxes represent 95% confidence intervals. Answers were given on a 5-point Likert scale ranging from 1 to 5. High values indicate a positive rating (e.g., helpful, easy, good, pleasant) and vice versa. There was a significant difference in responses to I3 regarding the easiness of revisiting specific sections within documents.

To test our hypotheses, we compared display space usage, measured as the percentage of display space covered by application windows in one-minute intervals. The average display space covered by application windows over the entire task in the control group was significantly higher than in the OG group ($t(18) = -4.991, p < 0.001$, 61% vs. 38%, see Figure 5.10). This difference was similarly pronounced for the maximum display usage with 71% control to 49% for the OG group, which is also statistically significant ($t(18) = -3.254, p = .004$).

Figure 5.11 shows heat maps of how the displays were used during the analysis session. To create these figures, the position and size of each window was aggregated for all users. Note how, in OG, the windows are concentrated on the two center screens, while the peripheral four monitors were used only occasionally.

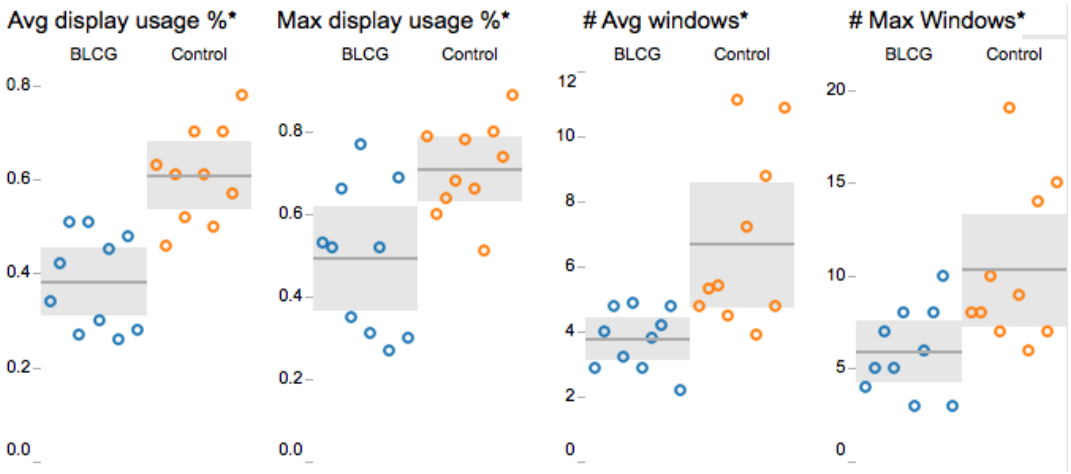
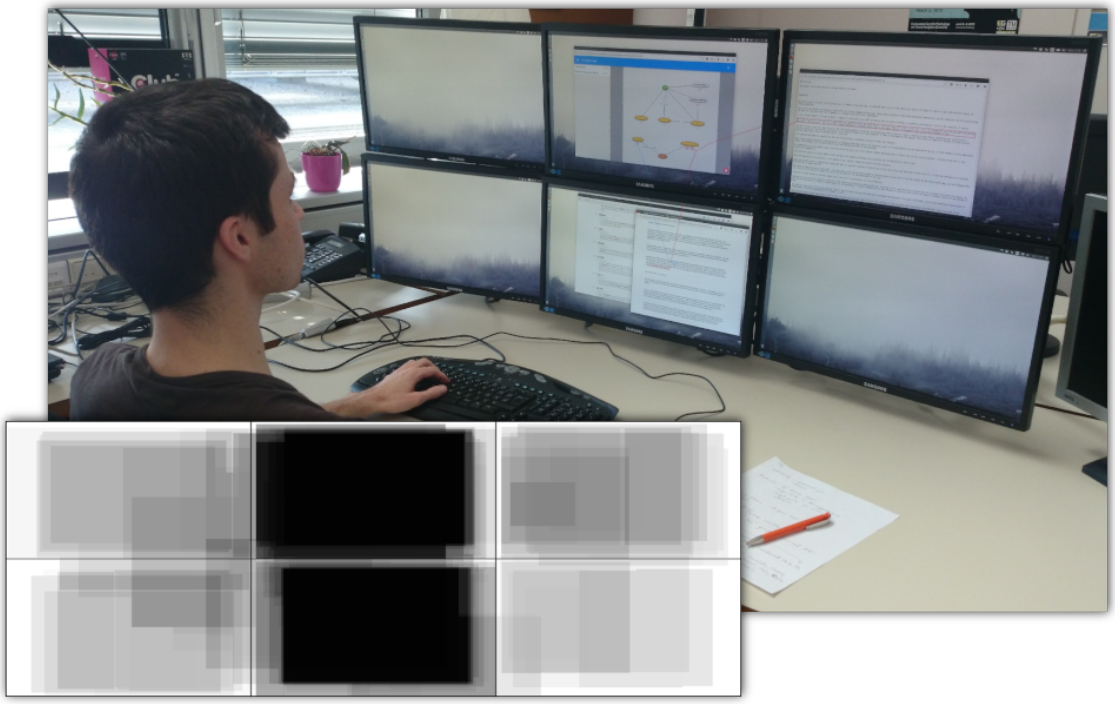


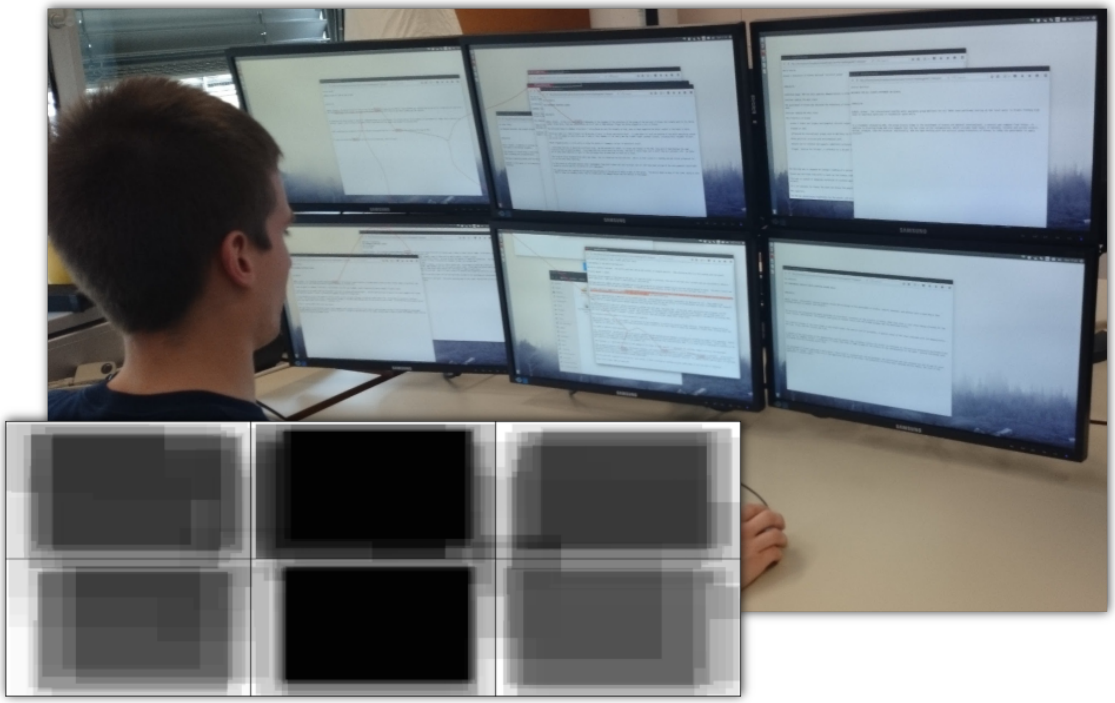
Figure 5.10: Usage of display space and window numbers were significantly higher in CC.

We found a considerable difference between the number of open application windows in the two conditions. The number of open application windows was significantly higher for CC than OG (6.7/3.8 average, $U = 12, p = .004$); see Figure 5.10. The maximum number of open windows was also significantly higher for CC (10.3/5.9, $t(18) = -2.860, p = .01$). CC users also conducted a significantly higher number of file queries (35.2/18.2, $U = 0, p < .001$). However, the number of opened files was similar in OG as in CC (31.1/29.3, $U = 44, p = .650$). The number of *distinct* files that were opened was almost equal (21.5/21.3). This implies that the difference was not caused by the number of visited documents, but by the way the document windows were managed. Most OG users only kept windows open occasionally. For instance, one user explained that *"I usually closed them right after usage to keep the space tidy."* CC users tended to keep documents with relevant content open for a longer time compared to OG users, or never closed them.

We also interviewed users about display arrangement strategies. While only four OG users mentioned a specific strategy how to arrange windows, seven users of CC were able to describe their window management strategies. We grouped interview responses into two different categories. The most popular strategy was to spatially group windows according to common concepts (described by two OG and four CC users). Two OG and three CC users partitioned their display into functional units, such as a main and peripheral area. However, these differences are only partially reflected in the questionnaire results. Overall, CC group rated the large display only slightly more pleasant to use (3.7 vs. 4.3), while, overall, the software provided was rated equally well by the two groups.



(a) OG condition



(b) control condition

Figure 5.11: Heatmap of the display usage in the OG and control condition showing the position and size of each window as recorded in one-minute intervals, aggregated for all users. Note that in the OG condition the windows are concentrated on the two center screens, while the other screens are used much less. Windows are much more distributed in the control condition.

In general, structuring approaches were diverse across all participants, but we could observe many commonalities. OG users created a more or less detailed concept graph, while CC users collected text snippets and notes in the provided Google Doc. Almost all users applied some groupings on their findings. While this is inherently supported by OG, all but one CC users also logically grouped blocks of text in their documents (e.g., by source document or abstract concepts, such as "bioterrorism" or "airport"). Additionally, users tried to maintain links to the original files. OG users had 6-33 file references in the graph. Similarly, all but two users in CC noted file names manually in the text files.

Many users mentioned in the interview that directly linking the source files to nodes or edges in OG was helpful. They revisited on average 7.4 files through such references. OG users rated *"It was very easy to find the relevant passages in the key documents again."* significantly higher than the control group (4.5/3.7, $U = 24, p = .037$). However, the question *"I had a very good overview of the documents I had already visited"* was rated low by both groups (2.8/2.4, $t(16) = -.339, p = .739$). Some users criticized that the search tool Recoll did not visually mark files that have already been opened. OG users explicitly noted that having numbers as file names made it hard to recall what the content of the particular file was. The node references therefore only showed up as numbers. With conventional web sources, OG showed a favicon or, if not available, the first letter of the sources domain name.

There was no difference in task performance between the two groups. All users made an effort to follow leads and to extract a potential terrorist plot — albeit not necessarily the correct one. Only few users identified elements of the ground truth plot (three OG users and four users of CC). The average number of opened documents out of the set of the 13 ground truth documents was low with 1.3 in OG and 1.6 in CC ($t(18) = -.468, p = .668$). Three OG users and four users of CC did not open any ground truth document at all. Users' subjective satisfaction with the outcome was stated similarly for the two groups, with 3.4 in OG and 3.0 in CC on average, on a 5-point Likert scale ($U = 43.5, p = .608$).

5.1.5 Multi-user Experiment

We ran the same task as before (section 5.1.3) with a pair of users and briefly report our observations. Our test users stated that, in their normal environment (sitting in the same office opposite to each other), they would use either a whiteboard or a GoogleDoc for sharing. They did not find it disturbing to work on the same computer and the same shared OG, as everyone could use a separate area. If desired, they shared findings in a

central area used by both. This behavior is also clearly visible on the heatmap shown in Figure 5.12, which shows the mouse pointer positions of each user, recorded in one minute intervals throughout the whole study.

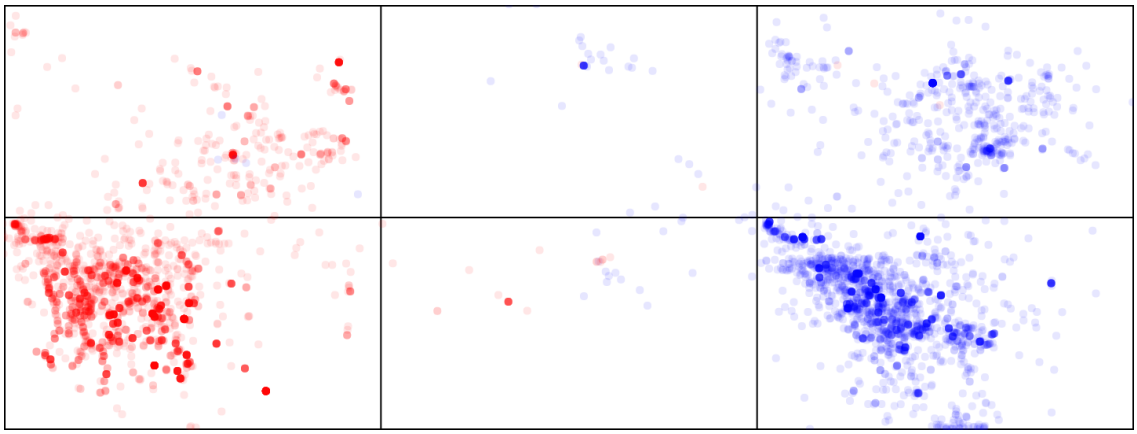


Figure 5.12: Heatmap of mouse positions during a study with two users. As they explained in the post-study interview the used the left and right sides as private workspace and the central monitors to share, show and collaborate.

Users liked dragging of websites and selections onto the OG (either to create new nodes or to attach document references to existing nodes). The OG was useful to see what the other person has already found. The OG was also very useful to show the other user something which they found earlier. They liked to be able to just look in the OG and click to open the document, without having to search the document again. Private areas (left and right) were utilized to perform individual search (Figure 5.13). The central displays were used as collaboration area to show documents to the other user, since the middle was the easiest location were both could read. However, they also noticed that sometimes no one felt responsible for windows on the shared space, and, as a result, unrelated windows were not closed, and therefore cluttered the available space.

Both users found it very helpful to get automatic links to newly opened documents, as it gave them awareness of related content already found by themselves, as well as by the other user. This allowed them to group together related documents very easy. One user also liked the indication of the other users link activity, as described in section 3.5.4, however, the other user did not notice them during the study. Furthermore, he would have liked to have the possibility to mark documents as being not useful, so that he or the other user do not have to read them again.

Finally, besides being used for intelligence analysis and other information foraging and sensemaking tasks, both users hypothesized that such a system could be very useful

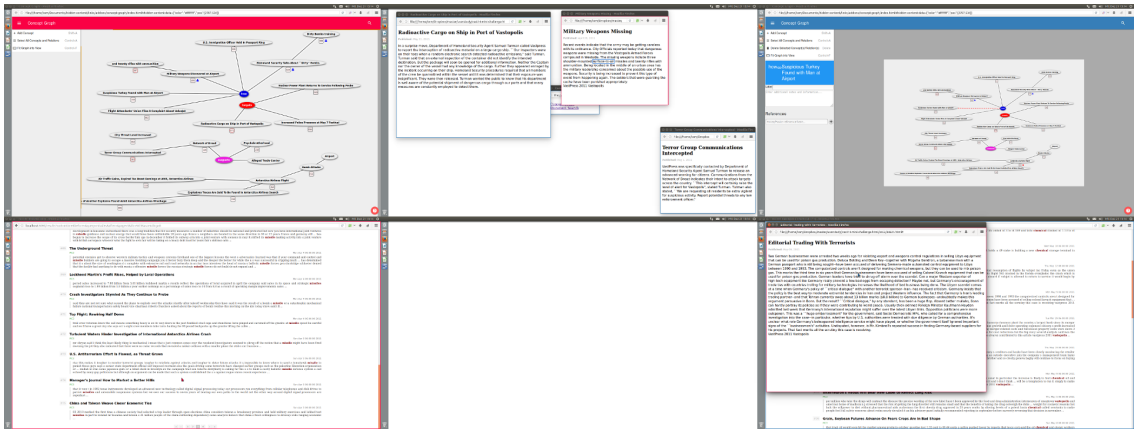


Figure 5.13: Screenshot of the final state of a desktop used by two users. The focii of the two users are indicated with red and blue color. Each user used their own graph and search window, and shared windows on the central monitors.

for programmers and requirement engineers to design, for example, class hierarchies or components and their connections, or, more generally, to perform any collaborative design task. For this, although, it would be beneficial to provide custom and domain specific symbols, shapes and connection types in the OG.

5.2 Guidance to Hidden Content

We conducted two exploratory user studies to evaluate our hidden content visualization techniques, as described in section 3.5, for three desktop scenarios of varying difficulty [Gey+14]. The scenarios involved up to three browser windows (including on map instance) for the first study and up to twelve web browser windows for the second. To gather meaningful feedback, a major part of the studies focused on an informal post test interview about the used hidden content visualization techniques.

5.2.1 Study Design

As baseline condition we used the **standard searching and highlighting** technique of the web browser (Firefox), synchronized across all browser windows, to isolate the effect of our visualization. Participants could mark (brush) words in the web browser and press a keyboard shortcut (Ctrl + F) to search the document for the marked word. The found terms use the default, colored box to highlight all occurrences. Pressing the Enter button repeatedly advanced to the next occurrence of the word within the same application. As an alternative method, participants could also type the search term into a text field. To search for the same term in a different application window, participants only had to switch the window and continue with the inspection of the highlighted occurrences. A new search term replaced the previous one. Switching windows was possible through standard operating system features.

We tested the baseline condition against our fully functional, real-time **hidden content visualization** implementation. The ability to interact with all types of links was provided, which either brought covered windows to the front or scrolled the regions outside the viewport into sight. The procedure to trigger a search was identical to the one used in the baseline condition.

The studies were conducted on a standard desktop computer (Intel Core i7-930 CPU @ 2.8GHz, AMD Radeon HD 7870 GPU, Ubuntu, Firefox) with a 22-inch monitor with a resolution of 1920x1080 pixels. The monitor was placed on a regular office desk and the users were seated approximately 70cm away.

Procedure

The studies were conducted as within-subject experiments over the described two conditions and three tasks for each condition. For each task, we measured task completion time and correctness.

During the first study, participants told the experimenter when they were ready to start with a task and reported their answer to the experimenter when they were finished. The experimenter measured the time for each task on a stop watch.

For the second study, we provided the participants with a tool to measure time and record answers (Figure 5.17). To start a trial, participants clicked on a button located in the master window on screen. The same window included check boxes for answering the questions and completing the trial. We automatically measured the time between the initiation and competition of a trial.

Prior to each task, participants were given a warm-up period, which allowed them to become familiar with the technique and the content of the application windows. After each condition, participants were required to assess their subjective satisfaction with the technique on a questionnaire containing six questions. After the hidden content visualization condition, we presented them with an additional questionnaire, comparing the individual approaches we employ for the different kinds of hidden content. Upon completion of the experiment, the participants were asked to take part in an unstructured interview.

5.2.2 Hypothesis

The goal of the user studies was to compare the effectiveness and efficiency of our hidden content visualization techniques and traditional search for finding hidden content in information analysis tasks. We formulated the following three hypotheses for this experiments:

[H1] Using the hidden content visualization leads to a faster retrieval of hidden data. Our techniques visualize all hidden regions that are placed within the boundaries of the screen and offers a preview for regions which are outside the screen. By interacting with these links, every hidden region can be accessed with one or two clicks. Thus, we expect our hidden content visualization to be faster than a sequential search through all hidden regions.

[H2] With the hidden content visualization, fewer errors are made. Because we visualize every occurrence of a search term, we expect users to miss fewer occurrences than by stepping through all windows and occurrences within windows sequentially.

[H3] Visualizing hidden content has a positive impact on understanding the spatial distribution of the data. Our hidden content visualization either shows the exact location of hidden regions or points towards the direction where they can be found. Thus, we hypothesize that it is easier for users to orient themselves within the data, if our visualization technique is used. As a consequence, we also expect it to be easier for users to locate data they have not yet explored.

5.2.3 Three Window Study

We recruited 18 participants from a local university (aged 24 to 36, 2 females). They were from the fields of computer graphics and visualization. 11 participants indicated that they had experience with visual data analysis.

Tasks and Apparatus

Users were asked to perform three information analysis tasks, with three desktop windows opened concurrently. All tasks dealt with the economic development of airports in two different geographic regions: Africa and Latin America. To generate two setups with equivalent complexity, we have altered the original data to include the exact same number of hidden and visible regions.

The default window setup is shown in Figure 5.14. We placed one browser window occupying the left side of the screen, which listed all airports in the chosen region, including their names and countries they belong to. On the lower right side, we placed a map application, which was centered over the area under investigation. On the top right side, we opened a second HTML document, which included the evolution of the busiest airports in the area over a period of four years. The only overlap, and thus possible source of covered regions was set up between the two windows on the right. The content of both browser windows was about ten times bigger than what could be presented on the screen.

The tasks were designed with different complexity levels in mind and were performed according to their complexity.

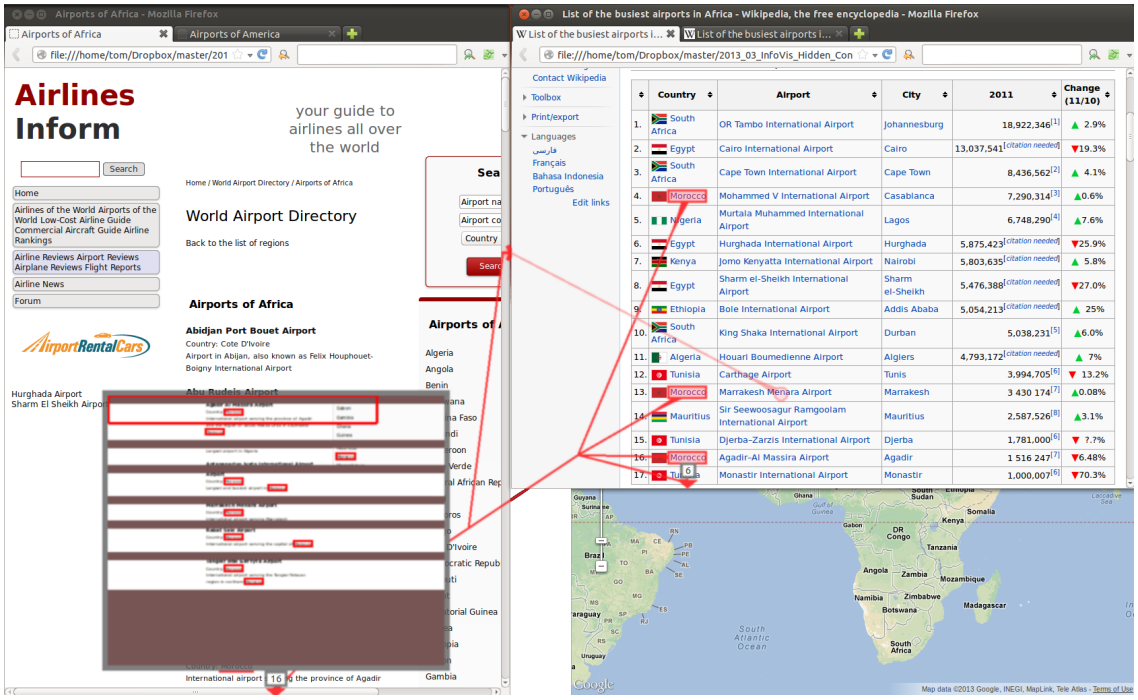


Figure 5.14: Window setup for the user study. One browser windows shows a list of airports in Africa or Latin America, a second browser window displays the evolution of the busiest airports in the area over four consecutive years, and a map centered on the same area is available.

Task 1 was designed as a simple search task with a variable number of hits and the requirement to take some information from the context area of the search results into account. It was to be carried out on the left browser window only, which contained a list of all airports in the area including some details about the airport. Users were asked to count the number of airports located in two given countries. As there was no one-to-one relationship between airport names and occurrences of the country name, a simply count of the number of highlighted regions or search hits was not sufficient to complete the task. Participants had to search the context region of the search terms. The only possibility for hidden content was outside the screen, as the window spanned the entire height of the screen and did not include any hits in the area scrolled away to the right.

Task 2 was designed as a data retrieval task in a structurally fixed setup. It was to be carried out in the top right window only, which listed the busiest airports in the area for the last three years. Users were asked to tell the experimenter which airports had an increase in the number of passengers per year for all shown years. To complete the task, users had to check if an airport was listed in all three tables, find the occurrences of the airport in the tables and check the passenger number for the given year. Thus,

the context information present in the table was highly important. For this task, hidden content was not only outside the screen, but also outside the window viewport. As there was only a single window needed for the task, no occlusions with the other window at the right took place.

Task 3 was designed to be the most complex task. It involved the combination of information from all three open windows. Users were asked to find all countries in the area, which are south of the equator (map application) and have an airport included in the list of the ten busiest airports (top right window). For each matching country, the users should count the number of airports found in the country (left window). As all three windows were involved in the task, content could be hidden due to being outside the screen, outside the viewport of one window, or covered by another window.

To remove the overhead of window reorganization from the measurements, participants were instructed not to change the window layout. To avoid learning effects due to knowledge of the data, each user completed all three tasks on both data sets, once with each technique. To reduce the influence of learning effects due to the repetition of tasks, the order of the conditions and the assignment of the condition to geographic region was counterbalanced.

Results

We measured the time users needed to complete each task, the correctness of the reported numbers, and subjective measures, which were given on a seven-point Likert scale. As a measure of correctness, we determined the deviation of the reported numbers from the actual number for task 1 and task 3 and the number of wrongly classified airports for task 2. As no category tested entirely positive for being normal distributed, all measures were evaluated using Wilcoxon Signed Rank tests ($\alpha = .05$). Timing results are illustrated in Figure 5.15 and questionnaire results are given in Figure 5.16.

Our analysis did not reveal a difference in completion time between the techniques for task 1 ($W = 29$, $p = .54$). We found a significant difference in completion time between the techniques for task 2 ($W = 125$, $p = .005$), and task 3 ($W = 123$, $p = .006$).

The average number of errors was very low for all tasks and techniques (traditional search: 0.03; guidance to hidden content: 0.04). There was no measurable difference to be found for error.

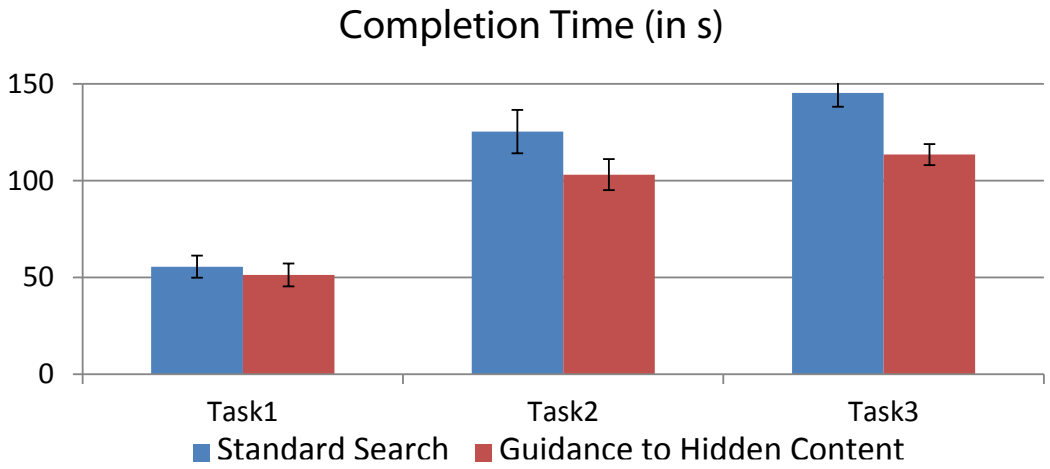


Figure 5.15: Mean completion time with standard error. While there is no difference for the simplest task (Task 1), guidance to hidden content perform significantly better for the more complex tasks.

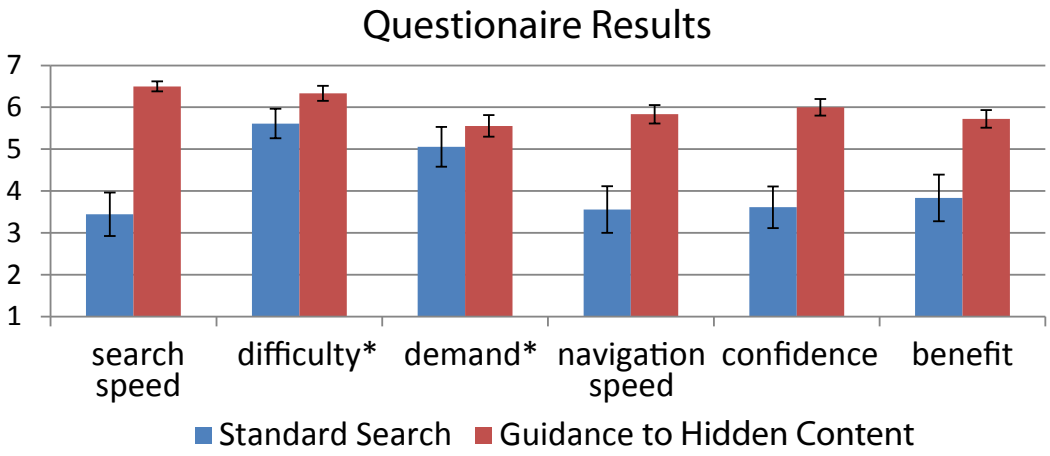


Figure 5.16: Mean questionnaire results given on a seven point Likert-scale. Higher results are better in all cases (* elements have been inverted).

We found a significant difference for the questionnaire items *subjective search speed* ('I could find the hidden content quickly.') ($W = 171$, $p < .001$), *subjective navigation speed* ('I could navigate to hidden content very quickly') ($W = 91$, $p = .002$), *subjective confidence* ('I am sure I did not miss any highlighted elements') ($W = 105$, $p = .001$). and *subjective benefit* ('The technique would be beneficial for my every day computer work') ($W = 90$, $p = .02$).

The difference in *subjective difficulty* ('It was very hard to find all hidden elements.') ($W = 71$, $p = .06$) and *subjective demand* ('The task was very mentally demanding') ($W = 27$, $p = .36$) were not statistically significant.

Observations and Feedback

All users quickly developed successful strategies using guidance to hidden content. For activities which involved checking regions outside the screen, the fastest participants accessed the required information directly in the preview pop-up without moving the viewport to the highlighted regions. This strategy could even be completed with the lowest zoom-level; most participants, however, zoomed in one level, which eased the reading of the preview.

If participants used the zooming feature too excessively, they sometimes became unsure which region they had already counted and so they had to start over. In the interview, they mentioned it would be useful to have even larger preview pop-ups to reduce the necessity of zooming. While concentrating on the preview, it does not matter if nearly all other content is covered. Two users had problems matching the viewport inside the preview pop-up and the real applications viewport, as, due to the compressed view, the aspect ratios do not match.

For regions outside the viewport, participants developed different strategies. If both regions outside the viewport and outside the screen appeared at the same time, participants most often used the preview only, because it also includes regions outside the viewport. If no regions outside the screen were present, users had to interact with the regions outside the viewport. About half of the users clicked on the regions to scroll them into sight; the other half hovered over the region and read the information directly from the superimposed preview. Four participants stated that they sometimes confused regions outside the viewport with regions inside other windows, and they would prefer the system treating these regions like regions outside the screen.

When regions were covered by other windows, only few participants accessed the covered content by hovering over the region or clicking on the region to bring the window to the front. Most participants clicked somewhere on the window to bring it to the front. In the interview, they stated that due to the simple window setup, it was easy to infer to which window the regions belonged to and it was simply faster to click on the window than moving the pointer to the region. When a location on the map was covered by the top right window, some participants did not explicitly access the map at all. If they only needed to check if a country was above or below the equator, the link to the covered regions was enough to answer the question as long the equator was not covered too.

Using guidance to hidden content, the least successful strategy was manual scrolling. Due to the delay introduced by the system, the rendered links always lag behind during smooth scrolling. This can be confusing and does not result in the ability to access content quickly. About half of the participants tried this strategy (at least during the warm-up period). All of them altered their strategy quickly. Most of them mentioned, that they believe that our visualization would also work well in conjunction with scrolling, if the delay could be reduced.

With a small amount of interesting content, participants sometimes where slower with marking a search term, initiating a new guidance process, opening the preview pop-up and zoom/scroll to the location than simply just scrolling there. Some users also mentioned that they think guidance to hidden content is not really useful for simple search tasks involving little content, but gets increasingly useful with complex tasks involving multiple windows.

With traditional search, participants often got lost while searching for elements. Most often they missed that the search had restarted at the beginning of the document. Even when the search advanced to the next occurrence, they sometimes were unsure, if they just saw the same content scrolled down a little, or if the entire content was new. In the interview, all participants confirmed that the guidance and the preview helped them a lot to get a better location awareness inside (large) documents and in this way reduced the mental load during search. Still, even with guidance, three participants found it sometimes difficult to orient themselves after jumping to a hidden location. They mentioned that some form of smooth scrolling would help.

Participants said that they would especially like to see the guidance feature to connect spreadsheet applications as well as code editors and documentations. Additionally, integrating a call hierarchy or references to a variable inside an IDE was considered as a useful extension.

Discussion

Based on the lower task time achieved with guidance to hidden content for task 2 and 3 and the higher subjective search and navigation speed, we conclude that *H1* is supported and guidance to hidden content leads to a faster retrieval of hidden content. The results also indicate that the usefulness of our technique increases with the complexity of the task. However, we assume that if the number of linked items on the screen increases too

much, the additionally introduced clutter and overlap between regions will lead to a worse performance. Due to the low error rate, we can not draw any conclusions about *H2*. Based on the participant feedback that guidance helps not to miss elements, it might be possible to confirm this hypothesis for more complex window setups. Because all participants stated that guidance to hidden content helped them to get a better location awareness inside documents, we conclude that *H3* is also supported.

The overall positive user feedback, which is also reflected in the questionnaire results, indicates that guidance to hidden content has a high chance of being used in real world situations, especially for complex information retrieval tasks. Due to the fact that the tasks focused on quickly gathering small pieces of information, participants did not need to read through text or other content. In cases where not only the highlighted regions are of interest to the user, the prominently drawn links might be disturbing or even cover important content. This fact was not captured in the study and techniques like fading out the visualization after some time might be needed in order to make the technique fully acceptable by users.

5.2.4 Twelve Window Study

Motivated by the results from the previous study, we improved the hidden content visualization techniques:

First, differentiating between out of viewport regions inside or outside the screen area showed to be not very useful and confused users, especially with many occurrences. Therefore, for the second study, we treated all out of viewport regions the same and did not show markers for regions outside the viewport, but still inside the screen area.

Second, visualizing individual covered regions also caused confusion. For this purpose, we introduced markers for windows containing hidden content, as described in section 3.5.2.

With these changes and improved system performance, we performed a second, more complex study with a larger number of windows. For this study, we recruited 18 participants from a local university (aged 20 to 37, 3 females) with a background in computer graphics and visualization. 10 participants indicated that they have experience with visual data analysis.

Tasks and Apparatus

Participants were asked to perform three information analysis tasks, with thirteen desktop windows opened concurrently. No restrictions where enforced during the test, i.e., users were allowed to move or to rearrange windows. All tasks dealt with properties of aircraft and airports. To generate two setups with equivalent complexity, we have altered two original data sets taken from Wikipedia to include the exact same number of hidden and visible regions for different properties.

The default window setup is shown in Figure 5.17. We placed a master browser window on the left side of the screen which was used to start a new trial and indicate answers. Twelve additional windows were randomly distributed across the whole screen, each approximately one third the size of the screen. The synchronized highlighting and guidance functionality was provided among all thirteen windows. Due to the large number of windows, participants were confronted with numerous overlapping windows and covered regions. Also, the shown documents were about six to ten times larger than the viewports they were displayed in.

We consciously avoided tabbed browsing, i.e., nested window management, in the study, since it would have made the design more complex, and we were primarily interested in testing the performance of our techniques with larger numbers of (partially) overlapping windows.

We designed the following tasks with increasing complexity:

Task 1 was designed as a simple information retrieval task testing the effectiveness of finding information. Participants had to find a single keyword in a subset of the open windows. They were additionally asked to take information from the context area of the search results into account. Participants should tell whether each of the six aircraft described within the documents is controlled using a yoke or a side-stick. For two aircraft, the information was unavailable and no answer was required.

Task 2 was designed as a more complex search task testing the efficiency of locating documents containing relevant information. Participants were asked to tell the experimenter which of two given keywords were contained within every of the twelve browser windows. The windows contained none, one, or both keywords.

Task 3 was designed as a research task testing the effectiveness of finding information. It required long content interaction, using three search terms, scanning through entire

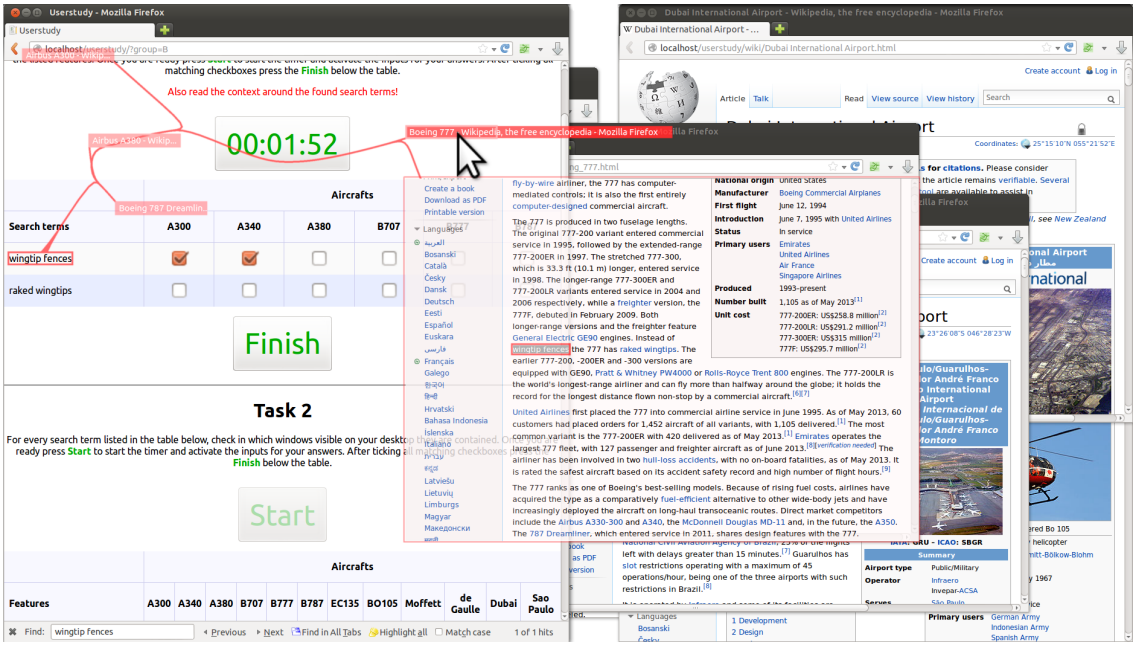


Figure 5.17: Window setup for the user study. Twelve browser windows are randomly arranged. Another window on the left, is used by the participants to start new trials and indicate their answers.

paragraphs, and reasoning. Participants were asked to find all aircraft equipped with a certain number of engines. The number of engines was indicated by a specific phrase. Participants had to read the context surrounding the phrase, as some documents contained the given phrase even though the aircraft had a different number of engines. Out of the six documents describing an aircraft, three contained the requested number of engines. For the remaining aircraft, participants where asked which of them have two more features installed, resulting first in two and finally one matching aircraft.

Participants completed all three tasks twice, once with each technique. To avoid learning effects due to knowledge of the data, all tasks where available with two different sets of keywords or features. To reduce the influence of learning effects due to the repetition of tasks, the order of the conditions and the assignment of the task sets was counterbalanced.

Results

We measured the time participants needed to complete each task, the correctness of the reported numbers, and subjective assessments, which were given on a seven-point Likert scale. As no category tested entirely positive for being normal distributed, all measures were evaluated using non-parametric tests. Wilcoxon signed rank tests ($\alpha = .05$) were

used for completion time, error rate, and the subjective task evaluation. The results comparing the individual approaches used by our hidden content visualization were analyzed using Kruskal-Wallis tests ($\alpha = .05$). Timing results are illustrated in Figure 5.18, and questionnaire results are provided in Figure 5.19 and Figure 5.20.

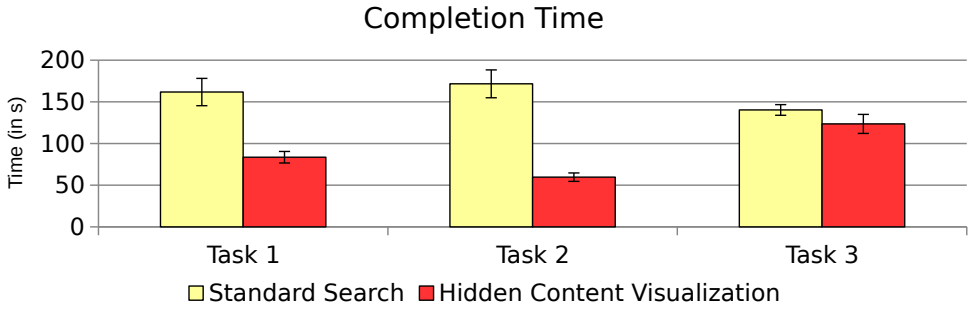


Figure 5.18: Mean completion time with standard error. Our hidden content visualization performs significantly better for tasks with a lot of unrelated content (Task 2) or only simple information retrieval (Task 1). For the research task (Task 3) the differences are not significant.

Our analysis revealed a difference in completion time between the techniques for Task 1 ($W = 141$, $p < .005$) and Task 2 ($W = 171$, $p < .001$). We found no significant difference in completion time between the techniques for Task 3 ($W = 80$, $p = .085$).

The average number of errors was very low for all tasks and techniques (traditional search: 0.05; hidden content visualization: 0.04). There was no measurable difference to be found for error.

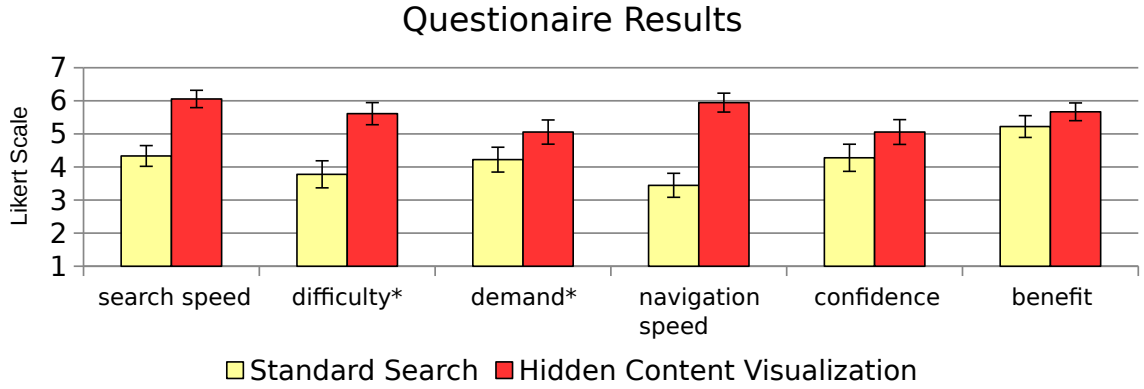


Figure 5.19: Mean questionnaire results given on a seven point Likert-scale. Higher results are better in all cases (* elements have been inverted).

We found a significant difference for the questionnaire items *subjective search speed* ('I could find the hidden content quickly.') ($W = 129$, $p < .005$), *subjective difficulty* ('It was very hard to find all hidden elements.') ($W = 106$, $p = .022$), and *subjective navigation speed* ('I could navigate to hidden content very quickly.') ($W = 135$, $p = .003$). The differences in *subjective demand* ('The task was very mentally demanding.') ($W = 70$,

$p = .07$), *subjective confidence* ('I am sure I did not miss any highlighted elements.') ($W = 49$, $p = .08$) and *subjective benefit* ('The technique would be beneficial for my every day computer work.') ($W = 38$, $p = .29$) were not statistically significant.

Observations and Feedback

Also in the this study, participants quickly developed successful strategies using our hidden content visualization, very similar to the first study.

If multiple windows containing scrolled-away content had large overlapping areas, some participants had problems to recognize which window each visualization belongs to. In the interview, they mentioned it would be useful to assign different colors to different windows or have the covered window marker also for partially covered windows.

In the hidden content visualization condition, no participant used the window manager to switch between open windows. Most of them mentioned that the hidden content visualizations enables them to locate target windows and regions faster than before. Two participants stated that they would like to have the indicators for found information inside the task-bar or window list, as they think a one-dimensional search within a linear list is faster than a two-dimensional search for highlights on the whole screen.

Participants said that enhancing the smart preview by showing only headings of the previewed document and expand or navigate to the sections content on demand would increase the acceptance as an everyday tool.

One participant tried to use scrolling within a see-through preview and stated in the interview the scrolling would be a useful extension to the preview.

Discussion

Based on the significantly lower task time achieved with our hidden content visualization for Task 1 and 2 and the higher subjective search and navigation speed, *H1* is supported again, and our hidden content visualization leads to a faster retrieval of hidden content. The results also indicate that the usefulness of our technique increases with the number of windows containing no relevant content. For Task 2, only half of the windows contained relevant content, which caused participants using standard search to check twice as many windows as required. Task 3 required far less navigation to different windows and,

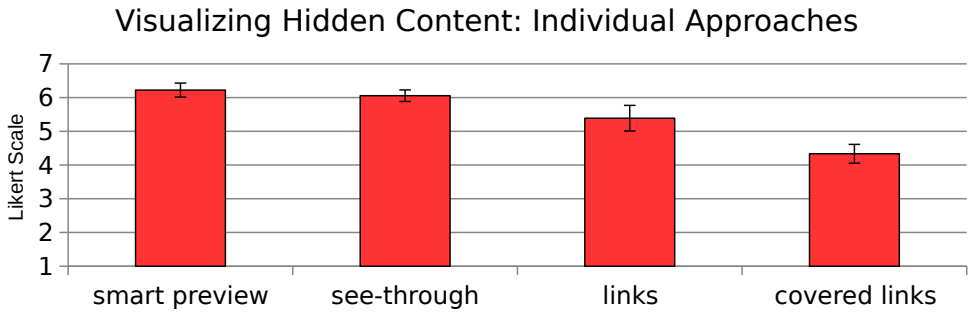


Figure 5.20: Mean questionnaire results for the different approaches combined our hidden content visualization given on a seven point Likert-scale. No significant difference was found in the data.

additionally, involved a reduced number of windows during each step. The largest part of the task required retrieving and combining information from the content. We believe that the major time-consuming activity for Task 3 was understanding and interpreting content instead of navigation and that this resulted in no significantly faster results for our hidden content visualization. Due to the low error rate, we still can neither accept nor reject *H2*. In line with the first study, all participants stated that our visualization helped them to get a better location awareness inside documents, again supporting *H3*.

The similar results to the first study and the overall positive user feedback, in particular for the smart preview and the see-through visualization of the smart links (see Figure 5.20), indicate that visualizing hidden content is a useful tool especially for complex information retrieval tasks possibly with a high amount of unrelated content.

6 Conclusion

We have presented a collaborative sensemaking environment, that is minimally invasive concerning current work practices and existing tools. Our environment spans a seamless desktop across many displays in a compute cluster, combined with web-centric application support. It supports foraging in web-applications and aggregating the findings explicitly in an observation graph (OG). By allowing visual links to span from the OG to evidence in the documents and vice versa, and, also to hidden content, we obtain a versatile tool that can accommodate a variety of work styles and supports gradual transition from perusing many documents to a more formal structuring of findings.

6.1 Spatial Organization

One of our key findings is that the use of *space to think* varies with the *tools to think*. This is equally true for individual users and collaborative work, although the details differ.

We have shown that users do not make use of even a moderately large, six-screen display setup to spatially organize documents, if they have an alternative for spatial organization of abstract concepts and quick retrieval of evidence. However, users do not stick to a single display either: Display space is used to visualize the abstracted information, to place all relevant tools side by side, and to compare multiple documents.

In our control condition using only visual links reaching across documents without supportive sensemaking tool, we could reproduce the findings of previous work: Users applied spatial strategies to organize a lot of open windows on a large display space. However, users of the OG did not perform any clustering or rough categorization of document windows. Instead, we observed this kind of behavior within the OG viewer itself. This implies that the spatial organization behaviors observed by Andrews et al. [AN12] are valid in general, but that source documents themselves are only used if no other, efficient abstract representation is available.

We also argue that the use of a sensemaking tool is preferable for other reasons: Documents often contain multiple concepts or relationships, and, therefore, a document can frequently not be sorted in a unique pile. It is trivial, however, to extract multiple concept nodes or relationships from one document and organize them in different categories. Also, a graph provides access to formalized, abstracted knowledge, which can be more easily revisited at a later time or communicated to others than an informal "pile" of documents.

We conclude that display space should be big enough to provide a "work zone" to efficiently look up and compare information, but that the need for display space to externalize mental models can be replaced by more compact abstract representations. This has immediate implications for knowledge workers: Instead of investing in large display setups and deal with their usability, technology, and performance limitations, it is better to invest in improved sensemaking tools.

While we only investigated the use of displays for information foraging based on text documents, we hypothesize that our results can be generalized to many visualization applications, where users will prefer succinct and meaningful abstractions to large display spaces. For example, Reda et al. [Red+15] and Yost et al. [YHN07] both argue that large displays are beneficial, because they can reduce zooming and panning. We agree that zooming and panning or switching between windows on a small screen are hindrances to exploration, but we argue that a moderately sized screen setup with enough space for the "work zone" and meaningful abstraction can be a powerful substitute for large display setups. In fact, OG can be considered a visual abstraction of the window layout, which itself is a visualization of the document content.

Of course, large displays have a distinct advantage: They can be filled with simple visualizations and do not require the development of meaningful abstractions for a wide variety of data types and usage scenarios. It is not even clear whether good abstractions exist for all relevant scenarios. However, the fact that even the largest display does not scale to an arbitrary dataset size provides motivation to investigate efficient abstractions independently of the display modality.

We also investigated the role of large displays with regards to collaboration. We could observe the establishment of personal territories [Tse+04]; however, the OG proved its value as sharing happened primarily through entering observations and link there. We conclude that more exploration of collaborative sensemaking using visual links provides exciting opportunities for future work.

6.2 Seamless desktop infrastructure

We have presented the first infrastructure that spans a seamless desktop across multiple computers without wrapping everything into a monolithic software architecture without any real applications. Instead, we rely on existing, mature web applications to manage the content and provide content-level synchronization across multiple computers and users. We achieve the illusion of a seamless desktop by the combination of software components for input redirection and output "redirection"; the latter in the form of web-browser instances that are manipulated by a custom software component. Our infrastructure is easy to set up and maintain, and leaves room for exploring new collaborative analysis strategies via arbitrary web application software.

In contrast to previous work, like SAGE2 [Mar+14] or SAViL [CN18], our framework does not monopolize input and output resources. We extend native system components, such as the window manager, web browser instances, mouse cursor and input focus. All features and behaviors of these system components, including enhancements from third-party desktop tools, are retained. This improves the chances that users can continue with their established work practices and adopt the new collaborative features as an enhancement rather than as a replacement of existing features. We believe that this deep integration with the underlying GUI is important for minimizing disruptive changes in the user experience and ensuring maximum productivity of the users.

6.3 Guidance to Hidden Content

We have presented techniques for visualizing search terms in areas of documents that are covered by other windows, outside the window's current viewport, outside the screen, contained in a minimized window, or in unopened files. We introduced smart previews that allow efficient exploration of content outside visible areas by providing a content-sensitive, space conserving, compressed preview of the whole document's virtual area, whereas smart links paired with a transparent overlay and click-to-show functionality allows for a fast navigation to covered content.

Future work should aim to reduce the visual clutter that may arise when a large number of regions are selected. We envision a combination of more sophisticated bundling algorithms, context-preserving routing [Ste+11], and smart fading of the links over time to further improve the situation.

Moreover, to improve temporal consistency and reduce lag, our system should be extended to track changes of relevant objects over time and incorporate temporal coherence into the layout planning. This would also allow us to avoid radical layout changes resulting from small changes (such as scrolling by a single line) in the underlying scene.

Finally, we want to pick up comments from study participants and plan to integrate our hidden content visualization techniques with an integrated development environment such as Eclipse, where search tasks and visiting all references and modifications of specific variables are crucial tasks in developing and debugging software.

Appendix

List of Acronyms

DOM Document Object Model [WHA98]. 43, 44, 57, 72

JSON JavaScript Object Notation [Bra14]. 14, 40, 41, 49, 56

MPX Multi-Pointer X [HT08]. 12, 14, 15, 21, 45–47, 49, 51

OG Observation graph. 2, 25–28, 37, 38, 45, 56, 57, 60–78, 93, 94, 118

Qt Qt cross-platform application development framework `https://www.qt.io/`. 14, 41, 46,

SDG Single-display groupware. 13

TCP Transmission Control Protocol. 39, 40, 49

UDP User Datagram Protocol. 41

List of Commands

This section lists all commands supported by the protocol described in section 4.1.1.

`CMD`
: Execute a command in a client application.

`REGISTER`
: Register as a client to the server. 41,

`RESIZE`
: Client notification to the server that its windows size has changed.

`SYNC`
: Synchronize the state of a client.

`CONCEPT-DELETE`
: Delete a concept.

`CONCEPT-NEW`
: Create a new concept.

`CONCEPT-UPDATE`
: Update properties of a concept.

`CONCEPT-UPDATE-REFS`
: Update the list of references of a concept.

`GET`
: Request a configuration value from the server.

`GET-FOUND`
: Response from the server as answer to GET with the actual value.

`SET`
: Set a configuration value of the server.

`DUMP`
: Dump the internal concept and link structure to the terminal.

`ABORT`
: Remove a link for a given keyword or all links.

`FOUND`
: Response to link request with the found regions.

`INFO`
: Sent from the server to clients to inform them about link requests from other clients. This allows clients to collect information about all currently active links, even if they are only active within other windows..

`INITIATE`
: Request the server to initiate a new link.

`UPDATE`
: Update the clients regions of a link for a given key keyword.

`CONCEPT-LINK-DELETE`
: Delete a link between two concepts.

`CONCEPT-LINK-NEW`
: Create a new link between two concepts.

`CONCEPT-LINK-UPDATE`
: Update properties of a link between two concepts.

`CONCEPT-LINK-UPDATE-REFS`
: Update the list of references of a link between two concepts.

`CONCEPT-SELECTION-UPDATE`
: Update the selected concepts and links.

`LOAD-STATE`
: Load links and graph from a previously saved file.

`REPLAY-LOG`
: Execute the user actions from a saved log file.

`SAVE-STATE`
: Save the current state of the system to a file. This includes the currently active links and the concepts, links and references in the graph.

`WM`
: Request the server to issue a command to the window manager. The following commands are available: `activate-window`, `open-url`, `windowownerchange`, `dragstart`.

Bibliography

[AEN10] Christopher Andrews, Alex Endert, and Chris North. "Space to Think: Large High-resolution Displays for Sensemaking." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. ACM, 2010, pp. 55–64. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753336. (Visited on 06/15/2015) (cit. on pp. 1, 11, 70).

[AM08] Saleema Amershi and Meredith Ringel Morris. "CoSearch: A System for Co-located Collaborative Web Search." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. New York, NY, USA: ACM, 2008, pp. 1647–1656. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357311. (Visited on 02/03/2016) (cit. on p. 12).

[AN12] Christopher Andrews and Chris North. "Analyst's Workspace: An embodied sensemaking environment for large, high-resolution displays." In: *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*. VAST '12. IEEE, 2012, pp. 123–131. DOI: 10.1109/VAST.2012.6400559 (cit. on pp. 1, 9, 11, 61, 93).

[Bau+04] Patrick Baudisch et al. "Collapse-to-zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content." In: *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, 2004, pp. 91–94. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029647. (Visited on 12/13/2013) (cit. on p. 7).

[BB05] Anastasia Bezerianos and Ravin Balakrishnan. "View and space management on large displays." In: *IEEE Computer Graphics and Applications* 25.4 (July 11, 2005), pp. 34–43. DOI: 10.1109/MCG.2005.92 (cit. on p. 23).

[BB09] Xiaojun Bi and Ravin Balakrishnan. "Comparing usage of a large high-resolution display to single or dual desktop displays for daily work." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. ACM, 2009, pp. 1005–1014. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518855. (Visited on 02/08/2013) (cit. on pp. 11, 70).

[BDB06] Anastasia Bezerianos, Pierre Dragicevic, and Ravin Balakrishnan. "Mnemonic rendering: an image-based approach for exposing hidden changes in dynamic displays." In: *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, 2006, pp. 159–168. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166279. (Visited on 02/15/2013) (cit. on pp. 1, 5).

[BE14] Sriram Karthik Badam and Niklas Elmqvist. "PolyChrome: A Cross-Device Framework for Collaborative Web Visualization." In: *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*. ITS '14. New York, NY, USA: ACM, 2014, pp. 109–118. ISBN: 978-1-4503-2587-5. DOI: 10.1145/2669485.2669518. (Visited on 02/04/2016) (cit. on p. 13).

[Ber+08] Michael Bernstein et al. "Information Scraps: How and Why Information Eludes Our Personal Information Management Tools." In: *ACM Transaction on Information Systems* 26.4 (Oct. 2008), 24:1–24:46. ISSN: 1046-8188. DOI: 10.1145/1402256.1402263 (cit. on p. 10).

[BG04] Patrick Baudisch and Carl Gutwin. "Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, 2004, pp. 367–374. ISBN: 1-58113-702-8. DOI: 10.1145/985692.985739. (Visited on 03/12/2013) (cit. on p. 6).

[Bie+93] Eric A. Bier et al. "Toolglass and magic lenses: the see-through interface." In: *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*. ACM, 1993, pp. 73–80 (cit. on p. 6).

[BNB07] Robert Ball, Chris North, and Doug A. Bowman. "Move to Improve: Promoting Physical Navigation to Increase User Performance with Large Displays." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. ACM, 2007, pp. 191–200. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240656. (Visited on 03/24/2016) (cit. on p. 11).

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3: Data-Driven Documents." In: *IEEE Transactions on Visualization and Computer Graphics*. InfoVis '11 17.12 (2011), pp. 2301–2309. DOI: 10.1109/TVCG.2011.185 (cit. on p. 56).

[BR03] Patrick Baudisch and Ruth Rosenholtz. "Halo: a technique for visualizing off-screen objects." In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, 2003, pp. 481–488. DOI: 10.1145/642611.642695 (cit. on p. 6).

[Bra+13] Lauren Bradel et al. "Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality." In: *International Journal of Human-Computer Studies* 71.11 (2013), pp. 1078–1088. DOI: 10.1016/j.ijhcs.2013.07.004 (cit. on p. 13).

[Bra14] Tim Bray. *The JavaScript Object Notation (JSON) Data Interchange Format*. Request for Comments 7159. Published: RFC 7159 (Proposed Standard). IETF, Mar. 2014. URL: https://tools.ietf.org/html/rfc7159 (cit. on pp. 40, 99).

[Byr99] Donald Byrd. "A Scrollbar-based Visualization for Document Navigation." In: *Proceedings of the ACM Conference on Digital Libraries (DL '99)*. ACM, 1999, pp. 122–129. DOI: 10.1145/313238.313283 (cit. on p. 7).

[CBF14] Olivier Chapuis, Anastasia Bezerianos, and Stelios Frantzeskakis. "Smarties: An Input System for Wall Display Development." In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*. CHI '14. New York, NY, USA: ACM, 2014, pp. 2763–2772. DOI: 10.1145/2556288.2556956 (cit. on p. 13).

[CC07] Christopher Collins and Sheelagh Carpendale. "VisLink: Revealing Relationships Amongst Visualizations." In: *IEEE Transactions on Visualization and Computer Graphics (InfoVis '07)* 13.6 (2007), pp. 1192–1199. DOI: 10.1109/TVCG.2007.70521 (cit. on pp. 8, 13).

[CGW14] Kris Cook, Georges Grinstein, and Mark Whiting. *VAST Challenge 2014: Mini-Challenge 1*. 2014. URL: http://www.vacommunity.org/VAST+Challenge+2014%3A+Mini-Challenge+1 (visited on 03/23/2016) (cit. on p. 63).

[Chu+14] Haeyong Chung et al. "VisPorter: Facilitating Information Sharing for Collaborative Sensemaking on Multiple Displays." In: *Personal Ubiquitous Com-*

puting 18.5 (2014), pp. 1169–1186. ISSN: 1617-4909. DOI: 10.1007/s00779-013-0727-2 (cit. on p. 11).

[CN18] Haeyong Chung and Chris North. "SAViL: cross-display visual links for sensemaking in display ecologies." In: *Personal and Ubiquitous Computing* 22.2 (2018), pp. 409–431. DOI: 10.1007/s00779-017-1091-4 (cit. on pp. 8, 11–14, 95).

[Cze+03] Mary Czerwinski et al. "Toward Characterizing the Productivity Benefits of Very Large Displays." In: *Proceedings of the IFIP TC.13 Conference on Human-Computer Interaction*. INTERACT '03. 2003, pp. 9–16. URL: http://www.idemployee.id.tue.nl/g.w.m.rauterberg/conferences/INTERACT2003/INTERACT2003-p9.pdf (visited on 02/08/2013) (cit. on p. 11).

[Dij59] Edsger W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/BF01386390 (cit. on p. 30).

[Dok10] Mark Dokter. "Synergy+ MPX: Towards Multi-User Interaction in Multi-Display Environments." Bachelor Thesis. Graz: Graz University of Technology, 2010. URL: http://www.icg.tugraz.at/project/deskotheque/publication/bachelor-thesis-dokter.pdf (visited on 01/27/2016) (cit. on pp. 15, 21, 47, 48).

[Don+06] Mira Dontcheva et al. "Summarizing personal web browsing sessions." In: *Proceedings of the ACM Symposium on User Interface Software and Technology*. UIST '06. ACM, 2006, pp. 115–124. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166273 (cit. on p. 10).

[DR02] A. Dieberger and D.M. Russell. "Exploratory navigation in large multimedia documents using Context Lenses." In: *Proceedings of the Hawaii International Conference on System Sciences (HICSS '02)*. 2002, pp. 911–917. DOI: 10.1109/HICSS.2002.994058 (cit. on p. 6).

[EGR91] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. "Groupware: Some Issues and Experiences." In: *Commun. ACM* 34.1 (1991), pp. 39–58. ISSN: 0001-0782. DOI: 10.1145/99977.99987. (Visited on 02/03/2016) (cit. on p. 13).

[ESS92] S.C. Eick, J.L. Steffen, and Jr. Sumner E.E. "Seesoft-a tool for visualizing line oriented software statistics." In: *IEEE Transactions on Software Engineering* 18.11 (1992), pp. 957–968. DOI: 10.1109/32.177365 (cit. on p. 6).

[FM11] Ian Fette and Alexey Melnikov. *The WebSocket Protocol*. Request for Comments 6455. Published: RFC 6455 (Proposed Standard). IETF, 2011. URL: https://tools.ietf.org/html/rfc6455 (cit. on p. 40).

[For+06] Clifton Forlines et al. "Multi-user, Multi-display Interaction with a Single-user, Single-display Geospatial Application." In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST '06. New York, NY, USA: ACM, 2006, pp. 273–276. ISBN: 978-1-59593-313-3. DOI: 10.1145/1166253.1166296. (Visited on 02/03/2016) (cit. on p. 12).

[Gey+14] Thomas Geymayer et al. "Show me the Invisible: Visualizing Hidden Content." In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. ACM, 2014, pp. 3705–3714. DOI: 10.1145/2556288.2557032 (cit. on pp. 3, 26, 28, 57, 79).

[Gey+17] Thomas Geymayer et al. "How Sensemaking Tools Influence Display Space Usage." In: *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2017, pp. 7–11. DOI: 10.2312/eurova.20171112 (cit. on pp. 4, 60, 72).

[Gey13] Thomas Geymayer. "Visual Links to Hidden Content." Master's Thesis. Graz: Graz University of Technology, 2013 (cit. on p. 28).

[GG00] Carl Gutwin and Saul Greenberg. "The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces." In: *Proceedings of the IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. WETICE '00. June 2000, pp. 98–103. DOI: 10.1109/ENABL.2000.883711 (cit. on p. 51).

[GLF13] Nitesh Goyal, Gilly Leshed, and Susan R. Fussell. "Effects of Visualization and Note-taking on Sensemaking and Analysis." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. New York, NY, USA: ACM, 2013, pp. 2721–2724. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481376. (Visited on 03/24/2016) (cit. on pp. 8, 9).

[Got07] David Gotz. "The ScratchPad: Sensemaking Support for the Web." In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. New York, NY, USA: ACM, 2007, pp. 1329–1330. DOI: 10.1145/1242572.1242834 (cit. on p. 10).

[Gri+11] Georges Grinstein et al. *VAST Challenge 2011: Mini-Challenge 3*. 2011. URL: http://hcil2.cs.umd.edu/newvarepository/VAST%20Challenge%202011/challenges/MC3%20-%20Investigation%20into%20Terrorist%20Activity/ (visited on 03/23/2016) (cit. on p. 73).

[Gru01] Jonathan Grudin. "Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. ACM, 2001, pp. 458–465. DOI: 10.1145/365024.365312 (cit. on p. 11).

[GS16] Thomas Geymayer and Dieter Schmalstieg. "Collaborative Distributed Cognition Using A Seamless Desktop Infrastructure." In: *Proceedings of the IEEE Virtual Reality Workshop on Immersive Analytics (IA)*. IEEE, 2016, pp. 7–12. DOI: 10.1109/IMMERSIVE.2016.7932375 (cit. on pp. 4, 19).

[Gus+08] Sean Gustafson et al. "Wedge: Clutter-free Visualization of Off-screen Locations." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, 2008, pp. 787–796. DOI: 10.1145/1357054.1357179 (cit. on p. 6).

[Har+95] Beverly L. Harrison et al. "Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. ACM, 1995, pp. 317–324. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223945. (Visited on 01/04/2014) (cit. on p. 6).

[HBW08] Raphael Hoffmann, Patrick Baudisch, and Daniel S. Weld. "Evaluating visual cues for window switching on large screens." In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. 2008, pp. 929–938. DOI: 10.1145/1357054.1357199 (cit. on pp. 8, 28).

[Hea95] Marti A. Hearst. "TileBars: Visualization of Term Distribution Information in Full Text Information Access." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. ACM, 1995, pp. 59–66. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223912. (Visited on 12/13/2013) (cit. on p. 6).

[HF01] Kasper Hornbaek and Erik Frøkjær. "Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+Detail Interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '01)*. ACM, 2001, pp. 293–300 (cit. on p. 7).

[HF03] Kasper Hornbæk and Erik Frøkjær. "Reading patterns and usability in visualizations of electronic documents." In: *ACM Transactions on Computer-Human Interaction* 10.2 (2003), pp. 119–149. ISSN: 10730516. DOI: 10.1145/772047.772050. (Visited on 03/12/2013) (cit. on p. 7).

[HKV95] Beverly L. Harrison, Gordon Kurtenbach, and Kim J. Vicente. "An Experimental Evaluation of Transparent User Interface Tools and Information Content." In: *Proceedings of the ACM Symposium on User Interface and Software Technology*. UIST '95. ACM, 1995, pp. 81–90. ISBN: 0-89791-709-X. DOI: 10.1145/215585.215669. (Visited on 01/04/2014) (cit. on p. 6).

[HS04] Dugald Ralph Hutchings and John Stasko. "Revisiting display space management: understanding current practice to inform next-generation design." In: *Proceedings of Graphics Interface*. Canadian Human-Computer Communications Society, 2004, pp. 127–134. URL: http://dl.acm.org/citation.cfm?id=1006058.1006074 (cit. on p. 11).

[HT08] Peter Hutterer and Bruce H. Thomas. "Enabling Co-located Ad-hoc Collaboration on Shared Displays." In: *Proceedings of the Ninth Conference on Australasian User Interface - Volume 76*. AUIC '08. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2008, pp. 43–50. URL: http://dl.acm.org/citation.cfm?id=1378337.1378346 (cit. on pp. 12, 14, 21, 46, 99).

[HW09] Danny Holten and Jarke van Wijk. "Force-Directed Edge Bundling for Graph Visualization." In: *Computer Graphics Forum (EuroVis '09)* 28.3 (2009), pp. 983–990. DOI: 10.1111/j.1467-8659.2009.01450.x (cit. on pp. 29, 30, 54).

[IF04] Edward W. Ishak and Steven K. Feiner. "Interacting with hidden content using content-aware free-space transparency." In: *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, 2004, p. 189. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029666. (Visited on 03/12/2013) (cit. on p. 6).

[IF09] Petra Isenberg and Danyel Fisher. "Collaborative Brushing and Linking for Co-located Visual Analytics of Document Collections." In: *Computer Graphics Forum* 28.3 (June 2009), pp. 1031–1038. ISSN: 01677055, 14678659. DOI: 10.1111/j.1467-8659.2009.01444.x (cit. on pp. 12, 13).

[Ise+09] P. Isenberg et al. "CoCoNutTrix: Collaborative Retrofitting for Information Visualization." In: *IEEE Computer Graphics and Applications* 29.5 (Sept. 2009), pp. 44–57. ISSN: 0272-1716. DOI: 10.1109/MCG.2009.78 (cit. on pp. 12, 13).

[Joh+01] Brad Johanson et al. "Multibrowsing: Moving Web Content across Multiple Displays." In: *Proceedings of the 3rd international conference on Ubiquitous Computing*. UbiComp '01. DOI: 10.1007/3-540-45427-6_29. Springer Berlin Heidelberg, Sept. 30, 2001, pp. 346–353. ISBN: 978-3-540-42614-1. URL: http://link.springer.com/chapter/10.1007/3-540-45427-6_29 (visited on 02/04/2016) (cit. on p. 13).

[Jos06] Simon Josefsson. *The Base16, Base32, and Base64 Data Encodings*. Request for Comments 4648. Published: RFC 4648 (Proposed Standard). IETF, Oct. 2006. URL: https://tools.ietf.org/html/rfc4648 (cit. on p. 41).

[Kan+11] Eser Kandogan et al. "How a Freeform Spatial Interface Supports Simple Problem Solving Tasks." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. New York, NY, USA: ACM, 2011, pp. 925–934. DOI: 10.1145/1978942.1979079 (cit. on p. 10).

[Kha+05] Azam Khan et al. "Spotlight: directing users' attention on large displays." In: *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, 2005, pp. 791–798. DOI: 10.1145/1054972.1055082 (cit. on p. 13).

[Kid94] Alison Kidd. "The marks are on the knowledge worker." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '94. ACM, 1994, pp. 186–191. ISBN: 0-89791-650-6. DOI: 10.1145/191666.191740 (cit. on pp. 8, 11).

[Kir95] David Kirsh. "The intelligent use of space." In: *Artificial Intelligence* 73.1 (1995), pp. 31–68. DOI: 10.1016/0004-3702(94)00017-U (cit. on pp. 8, 25).

[Lis+15] Lars Lischke et al. "Using Space: Effect of Display Size on Users' Search Performance." In: *Proceedings of the ACM Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '15. ACM, 2015, pp. 1845–1850. ISBN: 978-1-4503-3146-3. DOI: 10.1145/2702613.2732845. (Visited on 03/24/2016) (cit. on pp. 2, 11).

[Liu+14] Can Liu et al. "Effects of Display Size and Navigation Type on a Classification Task." In: *Proceedings of the ACM Conference on Human Factors in Computing*

Systems. CHI '14. ACM, 2014, pp. 4147–4156. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557020. (Visited on 03/25/2016) (cit. on p. 11).

[LL90] J. Chris Lauwers and Keith A Lantz. "Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '90)*. ACM Press, 1990, pp. 303–311. ISBN: 0-201-50932-6. DOI: 10.1145/97243.97301 (cit. on p. 13).

[LS87] Jill H. Larkin and Herbert A. Simon. "Why a Diagram is (Sometimes) Worth Ten Thousand Words." In: *Cognitive Science* 11.1 (Jan. 3, 1987), pp. 65–100. ISSN: 1551-6709. DOI: 10.1111/j.1551-6708.1987.tb00863.x. (Visited on 07/20/2016) (cit. on p. 25).

[Mal83] Thomas W. Malone. "How do people organize their desks?: Implications for the design of office information systems." In: *ACM Transaction on Information Systems* 1.1 (Jan. 1983), pp. 99–112. ISSN: 1046-8188. DOI: 10.1145/357423.357430 (cit. on pp. 8, 11, 71).

[Mar+14] T. Marrinan et al. "SAGE2: A new approach for data intensive collaboration using Scalable Resolution Shared Displays." In: *2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. 2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom). 2014, pp. 177–186. DOI: 10.4108/icst.collaboratecom.2014.257337 (cit. on pp. 14, 95).

[MPW06] Meredith Ringel Morris, Andreas Paepcke, and Terry Winograd. "TeamSearch: comparing techniques for co-present collaborative search of digital media." In: *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems, 2006.* TABLETOP '06. Jan. 2006, pp. 97–104. DOI: 10.1109/TABLETOP.2006.32 (cit. on pp. 12, 13).

[MT14] N. Mahyar and M. Tory. "Supporting Communication and Coordination in Collaborative Sensemaking." In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 1633–1642. DOI: 10.1109/TVCG.2014.2346573 (cit. on pp. 2, 9, 10).

[Ni+06] Tao Ni et al. "A Survey of Large High-Resolution Display Technologies, Techniques, and Applications." In: *Proceedings of the IEEE Virtual Reality*

Conference. Virtual Reality Conference. IEEE, 2006, pp. 223–236. DOI: 10.1109/VR.2006.20 (cit. on pp. 1, 11).

[Niw17] Ryosuke Niwa. *Selection API*. W3C Working Draft. W3C, June 2017. URL: https://www.w3.org/TR/2017/WD-selection-api-20170628/ (cit. on p. 44).

[PR94] Stephen Palmer and Irvin Rock. "Rethinking Perceptual Organization: the role of uniform connectedness." In: *Psychonomic Bulletin and Review* 1.1 (1994), pp. 29–55. DOI: 10.3758/BF03200760 (cit. on p. 8).

[Red+15] Khairi Reda et al. "Effects of Display Size and Resolution on User Behavior and Insight Acquisition in Visual Exploration." In: *Proceedings of the ACM Conference on Human Factors in Computing Systems*. CHI '15. ACM, 2015, pp. 2759–2768. DOI: 10.1145/2702123.2702406 (cit. on pp. 11, 94).

[RH11] Mikkel Rønne Jakobsen and Kasper Hornbæk. "Sizing Up Visualizations: Effects of Display Size in Focus+Context, Overview+Detail, and Zooming Interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. ACM, 2011, pp. 1451–1460. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979156. (Visited on 03/24/2016) (cit. on p. 11).

[RL04] Yvonne Rogers and Siân Lindley. "Collaborating around vertical and horizontal large interactive displays: which way is best?" In: *Interacting with Computers* 16.6 (Dec. 1, 2004), pp. 1133–1152. DOI: 10.1016/j.intcom.2004.07.008 (cit. on p. 13).

[Rob+04] George Robertson et al. "Scalable Fabric: flexible task management." In: *Proceedings of the ACM Working Conference on Advanced Visual Interfaces*. AVI '04. ACM, 2004, pp. 85–89. ISBN: 1-58113-867-9. DOI: 10.1145/989863.989874 (cit. on p. 11).

[SBD99] Jason Stewart, Benjamin B. Bederson, and Allison Druin. "Single Display Groupware: A Model for Co-present Collaboration." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. New York, NY, USA: ACM, 1999, pp. 286–293. DOI: 10.1145/302979.303064 (cit. on p. 13).

[Sch+02] M. C. Schraefel et al. "Hunter Gatherer: Interaction Support for the Creation and Management of Within-web-page Collections." In: *Proceedings of the*

ACM Confernce on World Wide Web. WWW '02. ACM, 2002, pp. 172–181. ISBN: 1-58113-449-5. DOI: 10.1145/511446.511469 (cit. on p. 10).

[SG86] Robert W. Scheifler and Jim Gettys. "The X Window System." In: *ACM Trans. Graph.* 5.2 (Apr. 1986), pp. 79–109. ISSN: 0730-0301. DOI: 10.1145/22949.24053. (Visited on 02/10/2016) (cit. on p. 46).

[SGL07] John Stasko, Carsten Görg, and Zhicheng Liu. "Jigsaw: Supporting Investigative Analysis through Interactive Visualization." In: *Proceedings of the IEEE Symposium on Visual Analytics in Science and Technology*. VAST '07. IEEE, 2007, pp. 131–138. DOI: 10.1109/VAST.2007.4389006 (cit. on pp. 2, 9).

[SK05] R. Saigal and A. Kumar. "Visual understanding environment." In: *Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries*. Proceedings of the ACM/IEEE-CS Joint Conference on Digital Libraries. JCDL '05. 2005, pp. 413–413. DOI: 10.1145/1065385.1065517 (cit. on p. 10).

[SR96] Mike Scaife and Yvonne Rogers. "External cognition: how do graphical representations work?" In: *International journal of human-computer studies* 45.2 (1996), pp. 185–213. DOI: 10.1006/ijhc.1996.0048 (cit. on p. 8).

[SS97] Kishore Swaminathan and Steve Sato. "Interaction Design for Large Displays." In: *interactions* 4.1 (Jan. 1997), pp. 15–24. DOI: 10.1145/242388.242395 (cit. on p. 13).

[Ste+11] Markus Steinberger et al. "Context-Preserving Visual Links." In: *IEEE Transactions on Visualization and Computer Graphics (InfoVis '11)* 17.12 (2011), pp. 2249–2258. DOI: 10.1109/TVCG.2011.183 (cit. on pp. 7, 8, 28, 30, 61, 95).

[Sto+12] A. Stoffel et al. "Document Thumbnails with Variable Text Scaling." In: *Computer Graphics Forum* 31.3 (2012), pp. 1165–1173. DOI: 10.1111/j.1467-8659.2012.03109.x (cit. on p. 7).

[Str+08] Marc Streit et al. "Navigation and Exploration of Interconnected Pathways." In: *Computer Graphics Forum (EuroVis '08)* 27.3 (2008), pp. 951–958. DOI: 10.1111/j.1467-8659.2008.01229.x. (Visited on 03/26/2010) (cit. on p. 8).

[Str+09] Marc Streit et al. "Caleydo: Connecting Pathways and Gene Expression." In: *Bioinformatics* 25.20 (2009), pp. 2760–2761. DOI: 10.1093/bioinformatics/btp432. (Visited on 02/02/2010) (cit. on p. 8).

[Suh+02] Bongwon Suh et al. "Popout Prism: Adding Perceptual Principles to Overview+Detail Document Interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*. ACM, 2002, pp. 251–258. DOI: 10.1145/503376.503422 (cit. on p. 7).

[SWS12] Markus Steinberger, Manuela Waldner, and Dieter Schmalstieg. "Interactive Self-Organizing Windows." In: *Computer Graphics Forum* 31.2 (2012), pp. 621–630. DOI: 10.1111/j.1467-8659.2012.03041.x (cit. on p. 6).

[TDR09] Philip Tuddenham, Ian Davies, and Peter Robinson. "WebSurface: An Interface for Co-located Collaborative Information Gathering." In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS '09. New York, NY, USA: ACM, 2009, pp. 181–188. ISBN: 978-1-60558-733-2. DOI: 10.1145/1731903.1731938 (cit. on pp. 12, 13).

[TIC09] M. Tobiasz, P. Isenberg, and S. Carpendale. "Lark: Coordinating Co-located Collaboration with Information Visualization." In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (Nov. 2009), pp. 1065–1072. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.162 (cit. on p. 13).

[Tse+04] Edward Tse et al. "Avoiding interference: how people use spatial separation and partitioning in SDG workspaces." In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW '04)*. ACM Press, 2004, pp. 252–261. DOI: 10.1145/1031607.1031647 (cit. on p. 94).

[VM12] Christophe Viau and Michael J. McGuffin. "ConnectedCharts: Explicit Visualization of Relationships between Data Graphics." In: *Proceedings of Eurographics/IEEE Conference on Visualization*. EuroVis '12. Vol. 31. 2012, pp. 1285–1294. DOI: 10.1111/j.1467-8659.2012.03121.x (cit. on p. 8).

[Wal+09] James R. Wallace et al. "Investigating teamwork and taskwork in single- and multi-display groupware systems." In: *Personal and Ubiquitous Computing* 13.8 (June 25, 2009), pp. 569–581. DOI: 10.1007/s00779-009-0241-8 (cit. on p. 13).

[Wal+10] Manuela Waldner et al. "Visual Links Across Applications." In: *Proceedings of the Conference on Graphics Interface (GI '10)*. Canadian Human-Computer Communications Society, 2010, pp. 129–136 (cit. on pp. 6, 8, 13, 39, 43, 53).

[Wal+11a] Manuela Waldner et al. "Display-adaptive window management for irregular surfaces." In: *Proceedings of the ACM Conference on Interactive Tabletops and*

Surfaces. ITS '11. ACM, 2011, pp. 222–231. DOI: 10.1145/2076354.2076394 (cit. on pp. 11, 70).

[Wal+11b] Manuela Waldner et al. "Importance-Driven Compositing Window Management." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. ACM, 2011, pp. 956–968. DOI: 10.1145/1978942.1979085 (cit. on p. 6).

[Wal11] Manuela Waldner. "WIMP Interfaces for Emerging Display Environments." PhD thesis. Graz, Austria: Graz University of Technology, June 2011 (cit. on p. 47).

[WHA98] WHATWG. *Document Object Model*. 1998. URL: https://dom.spec.whatwg.org/ (visited on 01/08/2018) (cit. on pp. 43, 99).

[Wri+06] William Wright et al. "The Sandbox for Analysis: Concepts and Methods." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '06. ACM, 2006, pp. 801–810. ISBN: 978-1-59593-372-0. DOI: 10.1145/1124772.1124890 (cit. on pp. 2, 9).

[WS11] Manuela Waldner and Dieter Schmalstieg. "Collaborative Information Linking: Bridging Knowledge Gaps between Users by Linking across Applications." In: *Proceeding of the IEEE Symposium on Pacific Visualization*. PacificVis '11. IEEE, 2011, pp. 115–122. DOI: 10.1109/PACIFICVIS.2011.5742380 (cit. on pp. 8, 11, 13).

[YHN07] Beth Yost, Yonca Haciahmetoglu, and Chris North. "Beyond Visual Acuity: The Perceptual Scalability of Information Visualizations for Large Displays." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. ACM, 2007, pp. 101–110. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240639 (cit. on pp. 11, 94).

[YW14] Jishuo Yang and Daniel Wigdor. "Panelrama: Enabling Easy Specification of Cross-device Web Applications." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. CHI '14. New York, NY, USA: ACM, 2014, pp. 2783–2792. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557199 (cit. on p. 13).

[Zha97] Jiajie Zhang. "The Nature of External Representations in Problem Solving." In: *Cognitive Science* 21.2 (1997), pp. 179–217. DOI: 10.1207/s15516709cog2102_3 (cit. on p. 8).

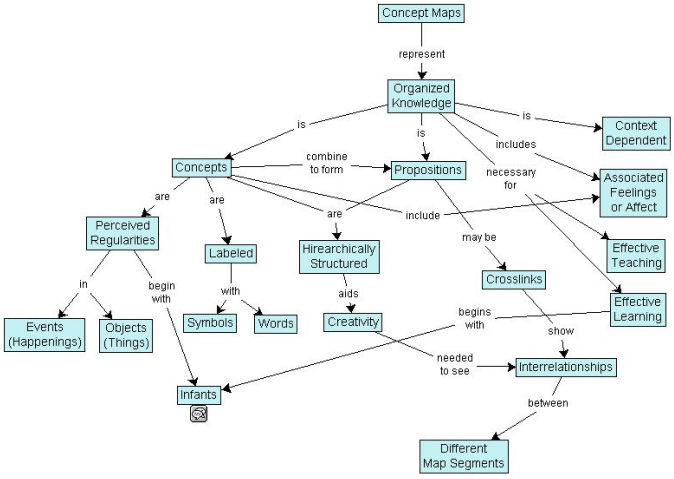
User studies supplemental documents

System description OG condition (section 5.1.1)

System Description

You will be asked to solve an intelligence task (details are provided later). We provide the following environment to analyze the data:

- You can use the entire display space to analyze your data
- You can use the file browser window to browse through the data
- You can use the recoll search tool to search within the provided documents
- You can use the concept graph tool to create a concept map (see figure below) about the data
    - In the concept graph window, click the middle mouse button to create a new concept (a node of the graph).
    - Alternatively, create a new concept by selecting text in a document and dragging the icon next to the browsers address input field onto the concept graph window.
    - You can freely move concepts inside the concept graph window.
    - Connect two concepts in the concept graph window by selecting both concepts and pressing [E] (or click “Relate Selected Concepts” in the side bar).
    - Alternatively, add a connection between two concepts by selecting both concepts in the concept graph window and afterwards selecting text in a document and opening the context menu. In the context menu select “Concept Graph Action” and use “Create Relation” to connect the selected concepts.
    - Add a link to a document to a concept or edge in the concept graph window by dragging the icon next to the browsers address input field onto a concept or relation.
    - Click on the desktop icon on the left upper monitor to open another window containing (a part of) the graph.



You will be able to play around with the system before starting the actual task until you feel comfortable.

System description control condition (section 5.1.1)

System Description

You will be asked to solve an intelligence task (details are provided later). We provide the following environment to analyze the data:

- You can use the entire display space to analyze your data
- You can use the recoll search tool to browse and search within the provided documents
- You can use the provided Google Doc to take notes
- You can use visual links to search for keywords across all open document windows
    - To start a link, mark the text you want to link in your open document window. Then open the context menu and select “keyword links” or press the “Ctrl” key.
    - To remove links, open the drop-down menu right of the browser address bar and click “abort all”.

You will be able to play around with the system before starting the actual task until you feel comfortable.

Graph Questionnaire (section 5.1.1)

Questionnaire

Study: _____ Condition: _____

Participant: _____

Please answer the questions below with a value from **1** (= I fully **disagree**) to **5** (= I fully **agree**).

G1. It was very easy to **add nodes** to the concept graph in the graph window

1 2 3 4 5

G2. It was very easy to **add edges** to the concept graph (i.e., to connect two nodes) in the graph

1 2 3 4 5

G3. It was very easy to **add nodes** to the concept graph from within the document

1 2 3 4 5

G4. It was very easy to **add edges** to the concept graph from within the document

1 2 3 4 5

G5. It was very easy to **remove nodes or edges** from the concept graph

1 2 3 4 5

G6. It was very easy to **arrange the nodes** of the graph

1 2 3 4 5

G7. It was very easy to **add references** to the concept graph

1 2 3 4 5

Graph Questionnaire (section 5.1.1)

I1. I had a very good **overview of the documents** I had already visited.

1 2 3 4 5

I2. It was very easy to **find and revisit key documents again**

1 2 3 4 5

I3. It was very easy to **find the relevant passages** in the key documents again.

1 2 3 4 5

D1. It was very easy to **find open documents** on the large display

1 2 3 4 5

D2. It was very easy to **manage (place, re-size...) multiple windows** on the large display

1 2 3 4 5

S1. It was very easy to **search** within the provided documents.

1 2 3 4 5

O1. I am very satisfied with the **outcome** of this session

1 2 3 4 5

O2. The **large display hardware** was very pleasant to use

1 2 3 4 5

O3. Overall, the **software** was very pleasant to use

1 2 3 4 5

Please rate how much the items listed below helped you to solve the task with a value from **1** (= not at all) to **5** (= very much).

Graph Questionnaire (section 5.1.1)

H1. Concept Graph				
1	2	3	4	5
H2. Search Tool				
1	2	3	4	5
H3. Large Display Space				
1	2	3	4	5
H4. Visual Links				
1	2	3	4	5

Links Questionnaire (section 5.1.1)

Questionnaire

Study: _____ Condition: _____

Participant: _____

Please answer the questions below with a value from **1** (= I fully **disagree**) to **5** (= I fully **agree**).

L1. It was very easy to **link** between documents

1 2 3 4 5

L2. It was very easy to **remove existing links** between documents

1 2 3 4 5

I1. I had a very good **overview of the documents** I had already visited.

1 2 3 4 5

I2. It was very easy to **find and revisit key documents again**

1 2 3 4 5

I3. It was very easy to **find the relevant passages** in the key documents again.

1 2 3 4 5

D1. It was very easy to **find open documents** on the large display

1 2 3 4 5

D2. It was very easy to **manage (place, re-size...) multiple windows** on the large display

1 2 3 4 5

Links Questionnaire (section 5.1.1)

S1. It was very easy to **search** within the provided documents.

1	2	3	4	5

O1. I am very satisfied with the **outcome** of this session

1	2	3	4	5

O2. The **large display hardware** was very pleasant to use

1	2	3	4	5

O3. Overall, the **software** was very pleasant to use

1	2	3	4	5

Links Questionnaire (section 5.1.1)

Please rate how much the items listed below helped you to solve the task with a value from **1** (= not at all) to **5** (= very much).

H2. Search Tool				
1	2	3	4	5
H3. Large Display Space				
1	2	3	4	5
H4. Visual Links				
1	2	3	4	5

Interview Questions Graph (section 5.1.1)

Interview

Study: _____ Condition: _____ Participant: _____

Task

Please provide the answers to the questions on the task description!

Concept graph

Explain all nodes and edges in the graph

Explain how you came up with the nodes and their connections

Describe how the information in the documents were turned into concepts (= nodes) and relations between concepts (= edges)

Was it easy to find information again that was referenced from the graph? (refer to questionnaire scores)

Was it easy to link information sources to the graph? (refer to questionnaire scores)

Display

How did you like the display setup? What did you like about it? What did you dislike?

Did you have a certain strategy how to use the available space?

Was it easy to find information again that was placed on the screen?

Did you keep a lot of windows open and distribute them on the screen? Why (not)?

Interview Questions Graph (section 5.1.1)

Comparison to everyday setup

What do you normally use to organize your information sources? (e.g., browser bookmarks, reference manager, ...)

Do you sometimes use mind mapping tools or something like that? Why not?

How would you solve a task like this on your normal workstation?

For which everyday task of yours would such an environment be beneficial? Would you use it?

Interview Questions Links (section 5.1.1)

Interview

Study: _____ Condition: _____ Participant: _____

Task

Please provide the answers to the questions on the task description!

Visual Links

Did you use the visual links (a lot)?

Do you think they were helpful to solve the task?

Describe in which situations you used the links?

Was it easy to find information again that you considered important before?

Display

How did you like the display setup? What did you like about it? What did you dislike?

Did you have a certain strategy how to use the available space?

Was it easy to find information again that was placed on the screen?

Did you keep a lot of windows open and distribute them on the screen? Why (not)?

Comparison to everyday setup

What do you normally use to organize your information sources? (e.g., browser bookmarks, reference manager, ...)

Interview Questions Links (section 5.1.1)

Do you sometimes use mind mapping tools or something? Why not?

How would you solve a task like this on your normal workstation?

For which everyday task of yours would such an environment be beneficial? Would you use it?

Task description pilot study (section 5.1.2)

Task Description

Note: This scenario and all the people, places, groups, technologies, contained therein are fictitious. Any resemblance to real people, places, groups, or technologies is purely coincidental.

In the roughly twenty years that Tethys-based GAStech has been operating a natural gas production site in the island country of Kronos, it has produced remarkable profits and developed strong relationships with the government of Kronos. However, GAStech has not been as successful in demonstrating environmental stewardship.

In January, 2014, the leaders of GAStech are celebrating their new-found fortune as a result of the initial public offering of their very successful company. In the midst of this celebration, several employees of GAStech go missing. An organization known as the Protectors of Kronos (POK) is suspected in the disappearance, but things may not be what they seem.

It is January 21, 2014, and you are called in to help law enforcement from Kronos and Tethys assess the situation and figure out where the missing employees are and how to get them home again. Time is of the essence.

You have one hour to provide answers for the following questions:

1) Who are the members of the Protectors of Kronos network:

   Who are the leaders?

   Who is part of the extended network?

   How has the group structure and organization changed over time?

   Where are the potential connections between the POK and GAStech?

You have the following data at your disposal:

- A map of Kronos
- A chart describing the local GAStech organization.
- A spreadsheet of GAStech employee records.
- Resumes and short biographies of many, but not all, of the GAStech employees
- Historical reports and descriptions of the countries involved
- Relevant current and historical news reports from multiple domestic and translated foreign sources, in text file format. Because these articles have come from multiple sources and original formats, some of them may contain corrupted characters, which is typical for this type of data.

You are free to use whatever tool you think is useful for solving the task.

Task description main study (section 5.1.3)

Task Description

Note: This scenario and all the people, places, groups, technologies, contained therein are fictitious. Any resemblance to real people, places, groups, or technologies is purely coincidental.

Intelligence analysts are looking for information related to potential terrorist activity in the Vastopolis area. News reports have been provided and it is up to you to identify any potential threats and give as much detail as possible on them, and be sure to include the documents you use as evidence.

Potential Threats: Identify any imminent terrorist threats in the Vastopolis metropolitan area. Provide detailed information on the threat or threats (e.g. who, what, where, when, and how) so that officials can conduct counterintelligence activities. Also, provide a list of the evidential documents supporting your answer. (Detailed answer)

Hidden Content Base Condition Questionnaire (section 5.2.1)

Participant _____ Synchronized Search

Please tick an answer from

1 (**totally disagree**) to **7** (**totally agree**):

Q1	I could find the hidden content very quickly.						
	1	2	3	4	5	6	7
Q2	It was very hard to find all hidden elements.						
	1	2	3	4	5	6	7
Q3	The task was very mentally demanding.						
	1	2	3	4	5	6	7
Q4	I could navigate to hidden content very quickly.						
	1	2	3	4	5	6	7
Q6	I am sure I did not miss any highlighted elements.						
	1	2	3	4	5	6	7
Q8	The navigation technique would be beneficial for my every day computer work.						
	1	2	3	4	5	6	7

Hidden Content Visualizations Questionnaire (section 5.2.1)

Participant _____ Guidance to Hidden Content

Please tick an answer from

1 (**totally disagree**) to **7** (**totally agree**):

Q1	I could find the hidden content very quickly.						
	1	2	3	4	5	6	7
Q2	It was very hard to find all hidden elements.						
	1	2	3	4	5	6	7
Q3	The task was very mentally demanding.						
	1	2	3	4	5	6	7
Q4	I could navigate to hidden content very quickly.						
	1	2	3	4	5	6	7
Q5	The highlighting technique introduced a high amount of visual clutter.						
	1	2	3	4	5	6	7
Q6	I am sure I did not miss any highlighted elements.						
	1	2	3	4	5	6	7
Q7	The highlighting technique was visually pleasing.						
	1	2	3	4	5	6	7
Q8	The highlighting technique would be beneficial for my every day computer work.						
	1	2	3	4	5	6	7

Q9	Preview Pop-ups make finding and navigating to scrolled content easier and faster.						
	1	2	3	4	5	6	7
Q10	The See-through technique helps finding and navigating to covered content.						
	1	2	3	4	5	6	7
Q11	Visual Links make finding and navigating to all types of content easier and faster.						
	1	2	3	4	5	6	7
Q12	Covered Links help finding and navigating to scrolled content inside the screen easier and faster.						
	1	2	3	4	5	6	7

Hidden Content Task Africa (section 5.2.3)

Task (Africa)

On your desktop, you can see three open application windows:

Google Maps: A map zoomed to Africa.

Firefox – list of the busiest airports in Africa: Rankings of the busiest airports in Africa for four consecutive years.

Firefox – list of airports in Africa: A list of all African airports and the countries they are located within.

Task 1

“Firefox – list of the busiest airports in Africa”

Which of the busiest airports in Africa had an increase in the number of passengers/year for all listed records?

Task 2

“Firefox – list of airports in Africa”

How many airports are located in Nigeria and Egypt respectively?

Task 3

All windows

Within a given time frame, we will ask you to count the number of airports in any country which is:

- in **Africa**
- **north of the equator** (countries along the equator do also count)
- which have an airport included in the list of the **10 busiest airports in Africa**

Tell the experimenter the number of airports of every country fulfilling the above requirements.

Tell the experimenter when you are ready to start.
You will get a clear signal when the time is over.

Hidden Content Task America (section 5.2.3)

Task (America)

On your desktop, you can see three open application windows:

Google Maps: A map zoomed to America south of the United States.

Firefox – list of the busiest airports in South America: Rankings of the busiest airports in South America for four consecutive years.

Firefox – list of airports in America: A list of all American airports south of the United States and the countries they are located within.

Task 1

“Firefox – list of the busiest airports in America”

Which of the busiest airports in America had an increase in the number of passengers/year for all listed records?

Task 2

“Firefox – list of airports in America”

How many airports are located in Mexico and Venezuela respectively?

Task 3

All windows

Within a given time frame, we will ask you to count the number of airports in any country which is:

- in **America** (South of the United States)
- **south of the equator** (countries with more then half of its area south of the equator do also count)
- which have an airport included in the list of the **10 busiest airports in South America**

Tell the experimenter the number of airports of every country fulfilling the above requirements.

Tell the experimenter when you are ready to start.
You will get a clear signal when the time is over.

Hidden Content Task Aircraft (section 5.2.4)