UNIVERSITY OF CALGARY

Lark: Using Meta-visualizations for Coordinating Collaboration

by

Matthew Andrew Tobiasz

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

CALGARY, ALBERTA

APRIL, 2010

UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Lark: Using Meta-visualizations for Coordinating Collaboration" submitted by Matthew Andrew Tobiasz in partial fulfillment of the requirements for the degree of Master of Science.

Dr. Sheelagh MST Carpendale
Supervisor
Department of Computer Science

Dr. John Daniel Aycock
Internal Examiner
Department of Computer Science

Mr. Gerald Marshall Hushlak
External Examiner
Department of Art

Dr. Michael Gordon Surette
External Examiner
Department of Microbiology and Infectious Diseases

Date

Abstract

Motivated by the increasing volume and complexity of real-world data, my research focuses on providing better software support for co-located collaborative information analysis. In particular, I consider *mixed-focus collaboration*—team work characterized by frequent changes in collaboration styles, which range from loosely coupled, individual work to closely coupled, group work. I identify three concepts—temporal flexibility, spatial flexibility, and scoped interaction—which play important roles in this type of work scenario. To this end, I design and implement Lark: a coordinated multiple view visualization environment where the relationships and connections between individual views are made explicit through an integrated meta-visualization. This provides both visual workspace awareness and explicit collaboration coordination points which provide group members with the freedom to work in concert or independently. The coordination of interactions can help facilitate mixed-focus collaborative work by supporting both individual and group work, and the transitions between these different types of collaboration.

Acknowledgements

I wish to thank the following people who supported and encouraged me during this thesis.

Thank you to my supervisor and mentor Sheelagh Carpendale. I am constantly inspired by your perspicacious view of the world. Your wisdom and encouragement has guided me in maturing academically, professionally, and personally, and I am absolutely honoured to have had this opportunity to work with you.

Thank you Petra Isenberg, for going first and making it look easy. Your support over the years has been unwavering and always positive. This project wouldn't have begun, nor would it have come as far as it has, without your involvement.

To all those people who at one time were members of the Interactions Lab and made it what it is, in particular: Fabrício Anastácio, Robin Arseneault, June Au Yeung, Christopher Collins, Marian Dörk, Mark Hancock, Helen He, Uta Hinrichs, Petra Isenberg, Ricardo Jota, Sean Lynch, Kimberly Mikulecky, Miguel Nacenta, Eric Pattison, Paul Saulnier, Stephen Voida, Mark Watson, and Torre Zuk.

Thank you to my biologist collaborators: Mike Surette, Chris Sibley, Cynthia Collins, and the rest of the members of the Surette Lab.

Thank you to Tim Burrell, Christopher Collins, Marian Dörk, Uta Hinrichs, Petra Isenberg, and Sean Lynch for the constructive comments on early drafts of this thesis.

Lastly, my appreciation and love goes out to my family and friends for all the encouragement, support, laughs, and love. To my family: Mom, Dad, Eric, Mary Lynn, and Anna. And my friends, many who I have already mentioned, plus: Megan Burnett, Tim Burrell, Dana Codding, Christopher Collins, David Custer, Ola Kowalewski, Carolyn Krahn, Melissa McDermott, Rachal Pattison, and Kris Read. I couldn't have done this, nor would of it been worth doing, without you.

To the memory of

Douglas Russell Cochran (1925 - 2007)

Publications

Materials, ideas, and figures from this thesis have appeared previously in the following publication:

Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale (2009). Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization)* 15(6): 1065–1072. doi: 10.1109/TVCG.2009.162

Table of Contents

List of Tables

List of Figures

List of Listings

Chapter 1

Introduction

In this thesis, I present research into a novel approach to coordinating interactions with information visualizations on shared digital workspaces. This approach makes use of an information visualization process model, or visualization pipeline, as the means to organize the collaborative workspace. This visualization pipeline is made visually explicit within the workspace through an integrated meta-visualization which supports both the awareness of how data visualization views are linked and the freedom for group members to work in concert or independently of one another. The research I present details the conceptual foundations behind this novel approach, in addition to the design and implementation of Lark, a system which embodies this collaboration and coordination concept.

To set the context for this research, I discuss how research on collaborative teamwork and information visualization plays an important role in addressing present day information challenges. In Section 1.2, I define the scope of my research by specifying the type of collaborative work I focus on and contextualizing my thesis within the larger research context. Next, in Section 1.3 I briefly introduce the milieu this research grew out of: a collaboration with a group of biologists which grounded and inspired the research reported in this thesis. To better explain the type of collaborative teamwork that this thesis focuses on, I present a fictitious collaborative scenario in Section 1.4. This scenario illustrates the characteristics of collaborative work that I am particularly interested in. In this scenario, I identify and define three important concepts often present in collaborative work. These concepts are used to structure the research challenges which I address in this thesis. I then itemize the specific

research challenges in Section 1.5. Lastly, in Section 1.6, I conclude the chapter with an outline of the organization of the remainder of the thesis.

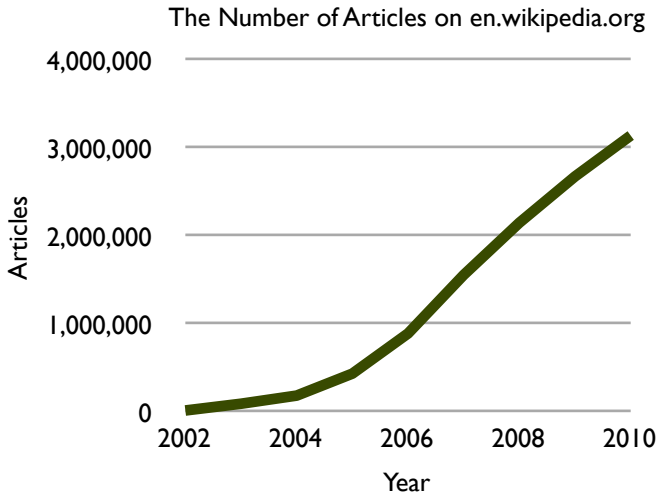
1.1 Motivation

The amount of accessible real-world information is continually growing. From the increasing number of articles in the publicly contributed online encyclopedia Wikipedia [2010a], to the expanding number of base pairs catalogued in the National Institutes of Health's open access genetic sequence database GenBank [Benson et al., 2000], the volume of available information is accelerating. Figure 1.1 illustrates the proliferation of these two examples from their inception to present day. The persistent growth rates observed in Wikipedia and GenBank are indicative of a pervasive trend seen in numerous other areas: more and more information is becoming readily available.

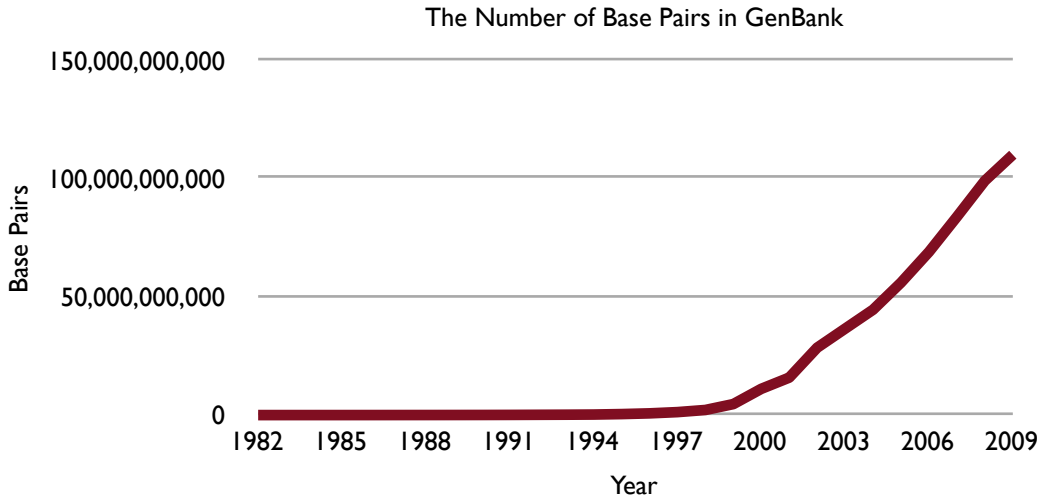
Analyzing and interpreting this burgeoning quantity of information is a difficult challenge. Utilizing the information processing capabilities afforded by information technology has helped in addressing this challenge; however, it is not a complete solution in itself. Computational resources excel at performing well defined information processing tasks, however, higher level analysis activities, such as "the crystallisation of knowledge from data" [Pattison and Phillips, 2001], are far beyond the capabilities of contemporary computer machinery. And so, people continue to play an integral role in high-level information-based activities.

The challenges present in the growing quantity of information persist at these higher level activities, and so, strategies for helping people manage these difficulties are needed. To this end, two important factors have been suggested: collaborative teamwork and information visualization [Thomas and Cook, 2005].

Collaborative teamwork involves groups of individuals working together on a common set of intersecting goals. It allows group members to combine their knowledge, potentially of different types and levels of expertise, enabling substantial coverage in both breadth and depth. For analytic activities, collaborative teamwork has the potential for leading to increased quality of solutions and discoveries, a testament to the Japanese proverb: *none of us is as smart as all of us* [Masum, 2002]. In Computer Science, research into supporting collaborative teamwork falls under the field



(a) The growth of total articles in English Wikipedia [Wikipedia, 2010c].



(b) The growth of catalogued base pairs in GenBank [National Center for Biotechnology Information, 2010]. Since the project's founding in 1982 this number has doubled approximately every 18 months.

Figure 1.1: Two examples of the increasing quantity of available information.

of Computer-Supported Cooperative Work (CSCW) which focuses on "groups of users—how to design systems to support their work *as a group* and how to understand the effect of technology on their work patterns" [Dix et al., 1993, p. 423]. Systems developed to support collaborative teamwork have the potential to help groups of individuals address challenges, such as the abundance of information, by empowering collaborative group activities with computational support that is efficient, effective, and satisfactory [Gutwin and Greenberg, 2000].

Information visualization is widely understood as "the use of computer-supported, interactive, visual representations of abstract data to amplify cognition" [Card et al., 1999, p. 7]. It has been suggested that:

> …visualization can amplify cognition: (1) by increasing the memory and processing resources available to the users, (2) by reducing the search for information, (3) by using visual representations to enhance the detection of patterns, (4) by enabling perceptual inference operations, (5) by using perceptual attention mechanisms for monitoring, and (6) by encoding information in a manipulable medium. [Card et al., 1999, p. 16]

Information visualization has the potential to be an effective means of working with large amounts of information. It is therefore increasingly recognized as being an important approach in supporting sense-making of the growing quantity of information. f

Both collaborative teamwork and information visualization are important factors in dealing with the information challenges of today. While considerable research is being conducted in both the CSCW and Information Visualization (InfoVis) communities, comparatively less research examines the interplay between them; this is especially true for co-located collaborative scenarios. At a high level, this thesis is motivated by the practical problem of how to best support the use of information visualizations during collaborative teamwork.

1.2 Research Context

In moving from a high-level characterization of my research problem to a concentrated set of research challenges, I now define the scope of my research. To this end, I narrow the general research scope by a set of constraints as to the type of collaborative work that I focus on. Then, I contextualize and further scope the thesis topic within the larger research context.

In this thesis, I am interested in collaborative work practices that make use of information visualizations during the data analysis process. I constrain my investigation to collaborative teamwork which meets the following three criteria: occurs in a synchronous co-located environment, uses a large multi-touch tabletop display, and involves small groups of people.

First, I restrict the type of collaboration to teamwork occurring in a *synchronous co-located* en-

vironment. That is to say, *synchronous*, occurring at the same time, and, *co-located*, occurring in the same physical space. A face-to-face group meeting would be an example of this type of collaboration. To provide additional clarity by way of antithesis, the polar opposite of synchronous co-located collaboration is *asynchronous remote* collaboration. Here, the collaboration is characterized by people working as a group from different spaces and at different times. Instances of collaborative work occurring outside of a synchronous co-located environment are beyond the scope of this thesis.

The use of information visualizations during collaborative teamwork sessions assumes that the collaborators have computational resources that are readily accessible when working together. One can imagine that the form factor of such devices can widely vary: from individual laptop computers or mobile computing devices, to a single high-performance computer workstation shared by the group. The second constraint is, therefore, to restrict the form factor of interest to *large multi-touch displays*. Large multi-touch wall and tabletop displays are a compelling form factor for supporting collaborative work as they offer new opportunities to support face-to-face collaboration, discussion, interpretation, and analysis around information displays, expanding the possibilities of desktop-based data analysis environments. *Large displays* allow multiple people to stand comfortably around a shared workspace with sufficient room for individual and group work. *Multi-touch* capabilities offer the opportunity for team members to manipulate both shared and individual instances of data representations concurrently. Combined *large multi-touch displays* offer new opportunities for supporting collaborative information visualization. Capitalizing on these affordances, I constrain my focus to the form factor of large multi-touch displays, especially digital tabletops.

Third, the size and technological capabilities of current large multi-touch displays are most ideally suited for use by small groups of people [Scott et al., 2003a]. Therefore, group size is constrained to small groups of approximately two to four people. This restriction is necessary in order to reliably provide a computing environment that offers rich digital interfaces and visualizations, where each collaborator can concurrently interact with the digital environment. A computing device with these affordances requires multi-touch input that is robust and at a high input resolution, in combination with a high resolution display, operating at a high pixel density, both of which must respond at interactive rates [Isenberg and Carpendale, 2007]. While these limitations in current hardware

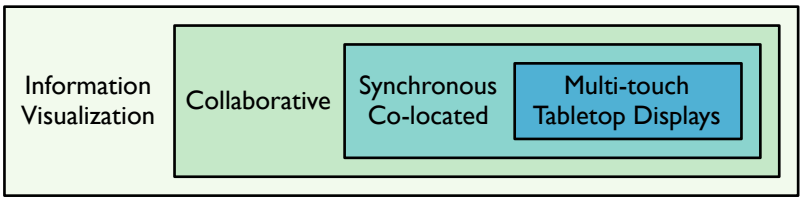


Figure 1.2: The research scope and context of this thesis. Note that this progressive nesting implies, for instance, that the research scope is focused on collaboration that happens within Information Visualization—not to all collaborative situations.

capabilities will undoubtedly be resolved in the near future, they present restrictions to the number of people that can be supported by the large multi-touch displays of today.

Combining these constraints, I focus my investigation of collaborative teamwork to supporting *mixed-focus collaboration*, which is known to occur under these conditions [Tang et al., 2006; Isenberg et al., 2008]. Mixed-focus collaboration is when group members frequently transition from loosely coupled, individual work to closely coupled, group work [Gutwin and Greenberg, 1998]. Here, *coupling* is defined as "the degree to which people are working together" [Baker et al., 2002].

The scope and context of this thesis research is illustrated in Figure 1.2. At a high level, this research falls under the field of InfoVis. Within InfoVis, my research focuses on the use of visualizations for collaborative data analysis tasks. In particular, collaborations in a synchronous co-located environment. My focus can be further narrowed towards exploring the use of multi-touch tabletop displays for collaborative work. The progressive refinement of my research scope is summarized in Figure 1.2.

1.3 Background

To ground my research in real-world data, tasks, and context, fellow student Petra Isenberg, Dr. Sheelagh Carpendale, and myself initiated a collaboration with a group of biologists led by Dr. Michael G. Surette. Dr. Surette's group is in the Faculty of Medicine, at the University of Calgary, researching microbiology and infectious diseases.

The research reported in this thesis was inspired by, and grounded in, discussions that came as a result of participation in Dr. Surette's research group. My colleagues and I were interested in investigating the collaborative processes already being practised within this group of biologists. Through

informal discussions and regular attendance of the biologists' weekly research meetings, we sought to gain insight into the group's collaboration dynamics, and use this insight as a starting point for our visualization research. In our discussions with this group of biologists, we were interested in how their collaborative practices could be facilitated by the use of digital tools, particularly for data analysis tasks. Our informal observations of the biologists' collaborative processes suggested that one of the most necessary pieces of technology needed to support collaboration were not novel data analysis tools, but rather an integrated data analysis environment with direct support for collaboration. While this thesis is set in a biological context, my primary focus is exploring ways to support collaboration.

1.4 Example Collaborative Scenario

To better identify the type of collaboration this thesis investigates, I describe an example scenario that follows a fictitious team of three biologists, Alice, Bob, and Carlos, working together to analyze clustered gene expression data from their latest experiment. A large number of genes and the complexity of the biological networks motivated the three to jointly analyze the data. This example illustrates the type of collaboration that is the focus of this research: small groups engaging in synchronous co-located collaborative data analysis. In this scenario digital tools are absent as the fictitious team makes use of paper-based visualizations and charts of their experiment results—a common approach employed by the biologists in Dr. Surette's research group. By focusing initially on a non-digital context, I identify the general characteristics of collaborative work practices that I wish to support in a digital context. The example highlights three important concepts often found in mixed-focus collaborative work: scoped interaction, temporal flexibility, and spatial flexibility. In the proceeding subsections, I define each of these concepts, identifying and explaining their occurrence during the collaborative scenario.

1.4.1 Parallel Work

The three analysts come together around a large table to begin their investigation. Each analyst has his or her own hard copy of the experimental results, printed out on sheets of paper. Since they do not know what to expect from the experimental data they broaden their coverage by exploring the data

Figure 1.3: Parallel work: Bob, Carlos, and Alice (from left to right) start by exploring the data individually.

in parallel, looking for interesting patterns individually (Figure 1.3). Alice wants to get an overview of the data. In front of her, she lays out numerous sheets of paper with the empirical results, hoping to get a general feel of the data. Bob takes a different approach. He wants to explore one particular part of the experimental data before looking at anything else. He locates the area of interest, and begins a careful examination of the data. Carlos decides to do a comparative exploration of data from a single gene under different experimental conditions, presented as a gene expression profile. He lays out the expression profiles next to one another, and begins his visual comparison.

This initial work partitioning outlines some of the pertinent collaboration aspects I am interested in. First of all, the large table allows each analyst to establish their own personal workspace in the larger shared work environment. This entails *spatial flexibility*, which is the unconstrained spatial organization and mobility of artifacts in and people in and around the work environment, allowing each analyst to organize him/herself in the workspace however he/she feels is most appropriate. For example, Alice spread out empirical data in front of her, seeking a high level overview. Carlos on the other hand, looked closely at a single gene under multiple conditions, placing the expression profiles next to one another for easy visual comparison. These two approaches to data analysis make use of

unconstrained physical arrangement of objects in the work environment to best match the chosen analysis approach. It is therefore important to be able to spatially organize objects in a customizable and dynamic fashion, thereby supporting different analysis styles.

Another salient concept is *scoped interaction*, defined as the extent of an action's effects as controlled by the performer of the action. In the scenario, all three analysts are able to concurrently interact with objects in their work environment. They moderate their interactions when appropriate, for instance, they can use their peripheral awareness of each other to flexibly ensure that each person's work can proceed unhampered. Scoped interaction means that all three analysts can work in parallel starting from their own unique analysis strategy, yet without disturbing the approach of his/her colleague.

1.4.2 Parallel and Joint Work

Carlos has found some interesting patterns in the data which he begins to investigate further. Alice notices that Bob is closely examining a section of data by glancing over at his workspace area. Intrigued by what she sees, she walks over to where Bob is working, gazing over his shoulder to get a better look at what he has found. Bob, welcoming Alice's assistance, turns the sheet of paper he is looking at so that Alice can have a better view. Bob explains what he has found by tracing through the analysis steps and the items in the data that brought him to this point in time. At each step in his explanation, Bob picks up the sheet of paper containing the experimental data he is currently referring to and passes it to Alice for closer inspection. The two begin discussing the significance of the data, as they examine the further details together. Figure 1.4 shows Alice and Bob working together, while Carlos works individually.

We see that as the analysis scenario progresses the collaborative cohesion has changed, from individual parallel work to individual parallel and joint group work. Alice and Bob are working closely together, while Carlos is in loosely coupled collaboration with his colleagues. Alice and Bob's transition in collaboration styles was heralded by Alice moving from her private workspace into closer proximity to Bob. Bob, noticing Alice's social cue shifts his current activities to include her, reorienting artifacts in his workspace so that they can be easily seen by Alice and explaining why he is

Figure 1.4: Parallel and joint work: Alice and Bob are discussing the data with one another while Carlos focuses on his own analysis.

interested in the piece of data that has his attention. This mobility of people and artifacts around the workspace (spatial flexibility), is important in facilitating changing collaboration styles. Bob's explanation involves temporally unravelling the analysis process he employed. This is an instance of *temporal flexibility*, which I define as, the unrestricted temporal ordering of activities. Bob is able to come back to his previous work and retrace his reasoning process, moving through activities he performed early in his analysis, before resuming his current activity. There is not specific temporal ordering to these activities, as such, Bob is able to perform them in whatever order he deems appropriate. This concept applies to group activities as well; they too have total control of the temporal ordering of their activities. Again, scoped interaction means that Alice and Bob can interact with items in the workspace without disturbing Carlos in his investigation.

1.4.3 Joint Work

At some point all three analysts decide to come together to see what they have found, closely discussing and negotiating their findings. Alice and Bob move over to where Carlos has been working, to have a better look at his results. Carlos reorganizes his workspace to be better viewed by the group

Figure 1.5: Joint work: Bob, Carlos, and Alice have moved to the same work region and are discussing the findings from their analysis.

and explains what he has been exploring, drawing attention to the items he finds most interesting. Upon identifying a common finding uncovered by all three analysts, the group moves over to Alice's workspace as she confirms her similar finding. As the discussion continues, the group moves about the workspace, verifying the conclusions from their analysis. Figure 1.5 shows Alice, Bob, and Carlos having moved to one area of the work space to discuss their findings together.

Again we see a shift in collaboration styles accompanied by movement of analysts and artifacts around the work space (spatial flexibility). The analysts revisit their previous activities to illustrate to the group the derived conclusions from their analysis (temporal flexibility). The discussion is supported by the physical arrangement of the data printouts in each analyst's work space (spatial flexibility).

1.4.4 Collaboration Scenario Synopsis

These scenarios present a possible example of the type of mixed-focus collaboration that I am interested in. The changes in collaboration styles throughout the duration of the collaborative analysis session are readily identifiable. So too are the presence of temporal flexibility, spatial flexibility, and

scoped interaction, which play an important role in facilitating the different collaboration styles and the transitions between styles. In deciding to support changing collaboration styles, I focus on these three germane characteristics, which are defined as follows:

Temporal Flexibility: interactions are flexible temporally if no constraints exist that require a specific sequencing or other temporal ordering of activities.

Spatial Flexibility: artifacts within, and people around the work environment, have spatial flexibility if no constraints exist as to the spatial organization of both artifacts and people.

Scoped Interaction: an interaction is explicitly scoped if information about the extent of an action's results is explicitly provided in the environment. This information provides an awareness of the actions of others, as well as control over one's personal actions. In this thesis I look at spatial scoping.

1.5 Research Challenges

From the example scenario, I have shown that during mixed-focus collaboration, there are frequent changes in the collaborative cohesion, ranging from loosely coupled, parallel work to closely coupled, group work. In this thesis, I investigate how to support these changing collaboration styles for synchronous co-located collaboration within a shared digital workspace. In particular, I address the following research challenges:

1. **How can we provide temporal flexibility for data analysis activities?** Recent evidence suggests that temporal flexibility among information analysis tasks is common practice among team workers [Isenberg et al., 2008]. I will investigate how to support this behaviour, particularly I will consider supporting concurrent interaction and not requiring any specific temporal flow of activities, thus allowing team members to follow their own unique analysis approaches.

2. **How can we provide spatial flexibility for visualizations and collaborators around the shared digital workspace?** Since changing collaboration cohesion in large workspaces is commonly accompanied by changing team member locations, it is important for artifacts in the workspace

to be mobile as well. In my research, I will provide a flexible approach to workspace organization that can allow team members to establish their own work areas [Scott et al., 2004], thus providing support for team members to coordinate their actions [Kruger et al., 2005].

3. **How can we provide scoped interaction?** Empowering team workers with the ability to work in parallel without interfering with each other's task is crucial for collaboration [Scott et al., 2003a]. My research investigates how each interaction within a digital workspace can be scoped in a manner that immediately and persistently informs all workers, thus supporting concurrent and asynchronous interaction.

1.6 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents an overview of previously published work related to my thesis research. This discussion begins with a definitions of visualization and the characterization of two unique visualization disciplines—Scientific Visualization (SciVis) and Information Visualization (InfoVis). Next, models for conceptualizing the visualization process are introduced and discussed in detail. This is followed by an overview of hierarchical data and a selection of representational techniques for visualizing this types of data. An overview of collaborative research from the perspectives of the Computer-Supported Cooperative Work, Tabletop, and InfoVis research communities is presented. The remainder of Chapter 2 outlines visualization research into coordinated multiple views techniques and meta-visualizations.

Lark, a system that facilitates the coordination of interactions with information visualizations on shared digital workspaces, is then introduced from three distinct perspectives: concept, interface design, and software architecture. Each of these perspectives are discussed in their own individual chapter, beginning with Chapter 3 which explains the high level concepts upon which Lark is built. Chapter 4 outlines the system's interface and interaction design as seen by someone working with the system. Chapter 5 presents an overview of the software architecture design and the prominent implementation details.

Chapter 6 revisits the example collaborative scenario introduced in Section 1.4, this time moving

to a digital context making use of the Lark system for collaborative data analysis. A discussion of the informal feedback on Lark from my biologist collaborators is presented, along with known limitations of the system, and an overview of exciting directions for future work. The thesis concludes with a reiteration of the research challenges and a summary of the research contributions which are made in this work.

Chapter 2

Related Work

The following chapter introduces the body of research that this thesis builds upon. I contextualize my thesis research contributions through this discussion, by providing the necessary background such that my contributions can be readily understood. At a high level, the chapter beings with theoretical considerations, moving towards the pragmatic issues by the end. Section 2.1 starts with a definition of *visualization*, the research field that this work falls under. I then characterize two visualization disciplines—Scientific Visualization and Information Visualization—and discuss how others have sought to differentiate these areas. In Section 2.2, I introduce how these two disciplines have conceptualized the process of visualizing data. I trace the history of two important process models—the *data flow* model and the *data state* model—in Section 2.3 and Section 2.4 respectively identifying how closely related these models are to one another and the often subtle refinements made during the evolution of each model. Work by Chi and Riedl [1998] introduced the most innovative contributions for the conceptualization of the information visualization process, and therefore, careful attention is paid to the important theoretical considerations identified in this work, as it provides much of the foundation upon which my research is based.

After the detailed discussion on visualization process models, the focus of this literature review shifts in Section 2.5 to a concrete example of visualizing a particular type of data: hierarchical data. A mathematical definition of hierarchical data and an overview of some of the common visualization techniques for representing this specific type of data are provided. Note, visualization techniques for

hierarchical data are not a contribution of this work; rather, they are used as an illustrative example of information visualization. Visualizing hierarchical data is used in later examples throughout this thesis, notably also in the Lark system, which is the focus of this work.

Moving from specific visualization techniques, in Section 2.6 I present an overview of research on facilitating collaboration amongst small groups of people in a synchronous co-located environment. I first look at foundational research from the Computer-Supported Cooperative Work community, defining the type of collaboration I am particularly interested in: mixed-focus collaboration. Next, I look at collaboration around interactive tabletop displays, the form factor which I use in my thesis research. The tabletop discussion is focused on a series of design guidelines for supporting effective co-located collaboration. This is followed by an overview of previous research on supporting synchronous co-located collaborative data analysis from the Information Visualization community. The concluding topic in the collaboration section is a synopsis of design guidelines for co-located collaborative information visualization systems, which is the design space my thesis research falls under.

Following the discussion on collaboration, in Section 2.7 I introduce how multiple representations of a common data set can be combined using a technique known as *coordinated multiple views (CMV)*. A brief overview of these techniques and previous work that investigates the design space is presented. Next, I introduce *meta-visualizations* which have been used to tackle some of the known complexity issues surrounding customizable CMV systems in Section 2.8. These topics relate to collaborative information visualization, as techniques from CMV in conjunction with meta-visualizations can be used to provide a foundation for creating a collaborative data analysis environment. It is in this direction that my own research extends. The primary concepts of this idea are presented in detail in Chapter 3.

2.1 Scientific and Information Visualization

Visualization is defined by Card et al. [1999, p. 6] as the transformation of data into visual representations, presented as interactive computer graphics with the goal of amplifying cognition. Within academe—as well as in medicine, entertainment, and industry—visualization has received growing

interest over the last thirty years, separating into two disciplines: Scientific Visualization (SciVis) and Information Visualization (InfoVis). Card et al. [1999, chap. 1] demarcates SciVis as generally focusing on scientific data that is physically based, while InfoVis deals with abstract non-scientific data. Fluid flow vectors from a hydrology simulation (scientific) and time series stock market data (non-scientific) are examples of the kind of data visualized in SciVis and InfoVis respectively. Another binary delineation between the two disciplines is that, generally, the data used in SciVis is intrinsically spatial [Card and Mackinlay, 1997]. On the other hand, in InfoVis the data often is without a spatial component and it is up to the visualization designer to assign a spatialization where appropriate [Munzner, 2002]. While these distinctions are a good starting point, on further scrutiny they can appear vague and contradictory at times [Tory and Möller, 2004]. Visualized data which identifies with InfoVis is often scientific in nature. (For example, visualizations of phylogenetic trees used in evolutionary biology to illustrate the relationships between species [Gregory, 2008] as investigated by Munzner et al. [2003].) Furthermore, it is unclear where visualized scientific data that does not have a spatial component, nor physical in nature, should be classified based on this binary characterization [Tory and Möller, 2004].

For a more robust classification of SciVis and InfoVis, we look to the model–based visualization taxonomy proposed by Tory and Möller [2004]. The taxonomy looks to the assumptions made about a visualization's *design model*, rather than the type of data to be visualized, in classifying a certain visualization technique. The design model is the set of assumptions made by the visualization designer when creating the visualization algorithm, and these assumptions influence the design decisions made and thereby impact the end visualization. For example, a surface topography might be stored as set of discretely sampled points (the data type), however the visualization algorithm interpolates this data into a representation that is a continuous surface. Here, the design model makes the assumption that, although the data is discrete, the *object of study* is continuous, and therefore, the visualization should represent it as such.

At a high level, Tory and Möller's taxonomy uses two main criteria in classifying data models: is the object of study discrete or continuous, and what amount of flexibility—given, constrained, or chosen—was afforded to the visualization designer in deciding the display attributes (such as spatial-

	Given	Display Attributes Constrained	Chosen
Continuous	Images (e. g., medical) Fluid/gas flow, pressure distributions Molecular structures (distributions of mass, charge, *etc.*) Globe—distribution data (e. g., elevation levels)	Distortions of given/ continuous ideas (e. g., flattened medical structures, 2D geographic maps, fish-eye lens views) Arrangement of numeric variable values	Continuous (high-dimensional) mathematical functions Continuous time-varying data, when time is mapped to a spatial dimension Regression analyses
Discrete	Classified data/images (e. g., segmented medical images) Air traffic positions Molecular structures (exact positions of components) Globe—discrete entity data (e. g., city locations)	Distortions of given/ discrete ideas (e. g., 2D geographic maps, fish-eye lens views) Arrangement of ordinal or numeric variables values	Discrete time-varying data, when time is mapped to a spatial dimension Arbitrary entity-relationship data (e. g., file structures) Arbitrary multi-dimensional data (e. g., employment statistics)

Table 2.1: The high-level visualization taxonomy [from Tory and Möller, 2004, p. 154], which uses two main criteria in classifying data models: is the object of study discrete or continuous, and what amount of flexibility—given, constrained, or chosen—was afforded to the visualization designer in deciding the display attributes. SciVis is traditionally placed in the top right corner of the matrix, with InfoVis in the bottom left.

ization, timing, colour, transparency, *etc.*). Based on these two criteria the traditional definitions of SciVis and InfoVis fall on opposing corners of the two-dimensional matrix illustrated in Table 2.1. Traditionally, SciVis occupies the top-left of the matrix and InfoVis the bottom-right. The typical object of study for SciVis is continuous, where the display attributes are given, such as spatialization. In InfoVis, the object of study is typically discrete, where the visualization designer has more freedom to choose the display attributes. Moreover, the taxonomy nicely illuminates the relatedness of these two fields within the continuum.

2.2 History of Visualization Process Models

Both the SciVis and InfoVis communities are interested in creating visual representations of data, whatever the type of data may be. To better understand this visualization process—the transformation of data into interactive computer graphics—it is useful to deconstruct the complex whole into its constituent parts. Both visualization communities have sought to do just this, creating models of the visualization process. These models have provided clarity ranging from conceptual, helping researchers think about visualization [dos Santos and Brodlie, 2004], to pragmatic, forming the architectural foundation for the implementation of visualization applications [Duke et al., 2006]. The two important visualization process models are the *data flow* and *data state* models [Chi, 2002]. The following two sections present an overview of both models, tracing the origins of each and following their maturation over time. The evolution of these process models are generally incremental, with only subtle refinements in each iteration. Since my research relies heavily on a visualization process model, I present a detailed discussion of the conceptualization of the visualization process and identifying how these ideas have evolved into formalized models.

2.3 Data Flow Model

I begin this discussion of visualization process models with the *data flow model* which originated in the SciVis community. The data flow model is a visualization process model where a series of data-processing stages progressively transforms raw numerical data into interactive computer graphics. The process is described using a pipeline metaphor, and hence visualization process models are often simply referred to as *visualization pipelines*.

The data flow model was first proposed by Haber and McNabb [1990], who built from earlier work on interactive graphics systems by Haeberli [1988] and visualization process formalizations by Upson et al. [1989]. In this section, I present a chronological history of the data flow model, beginning with work by Haeberli from the late 1980's on the ConMan system. The data flow model has proven to be an important contribution to conceptualizing the visualization process. Nearly two decades later, it continues to receive attention and use in the SciVis community [Chi, 2002; dos Santos and Brodlie, 2004; Duke et al., 2006].

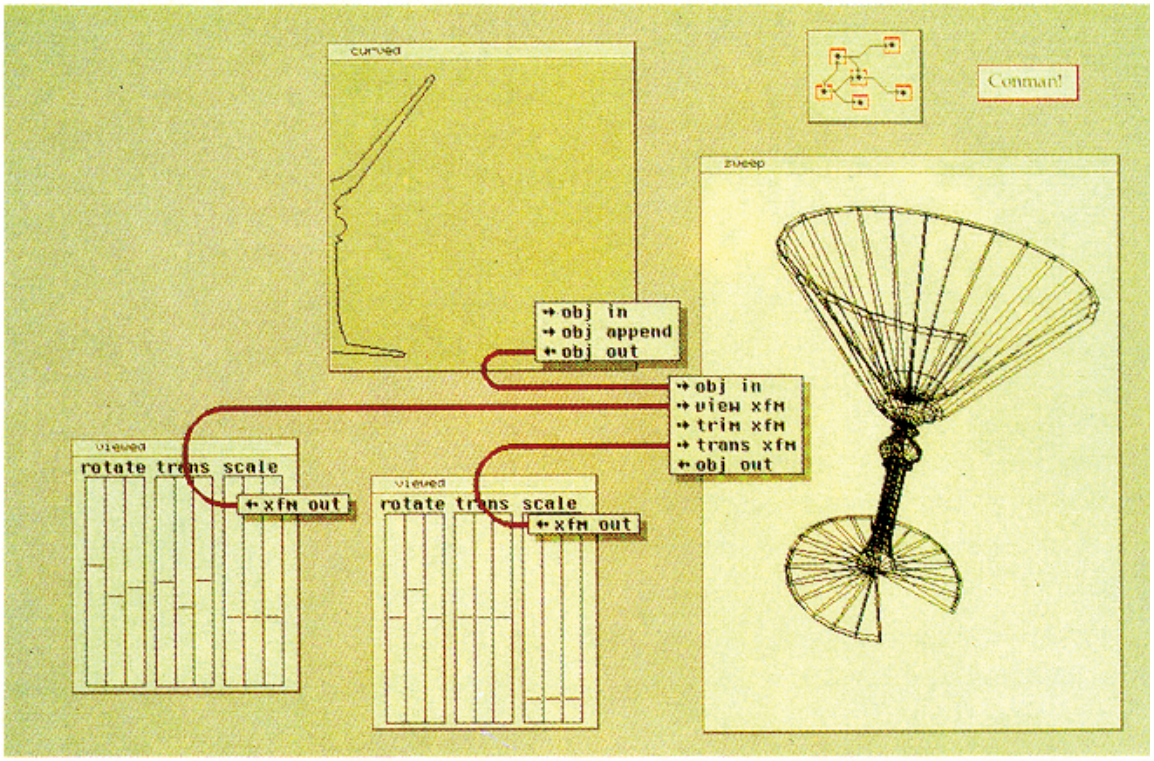


Figure 2.1: An example of the original ConMan graphical user interface, showing a simple application [from Haeberli, 1988, p. 105]. Visualizations are constructed using a high level visual programming language, organized as a directed acyclic graph. Data *flows* from the outputs of one component into the inputs of another component, creating a pipeline of interconnected components. The pipeline accumulates in the production of an image, shown here in the window on the right. © *1988 ACM, Inc. Included here by permission.*

2.3.1 ConMan: A Data Flow Metaphor for Visualization:

The first known instance of articulating the visualization of data through the use of a data flow metaphor was reported by Haeberli [1988] in the ConMan system. ConMan, or Connection Manager, is a high level visual programming language in which interactive graphics can be constructed by graphically creating and modifying a series of interconnected components. The interface for the system is organized as a directed acyclic graph, in which vertices are components and edges are connections between components. An example of the original interface is shown in Figure 2.1. Data *flows* from the outputs of one discrete component and into the inputs of another component. This *pipeline* of interconnected components continues, accumulating in the production of an image.

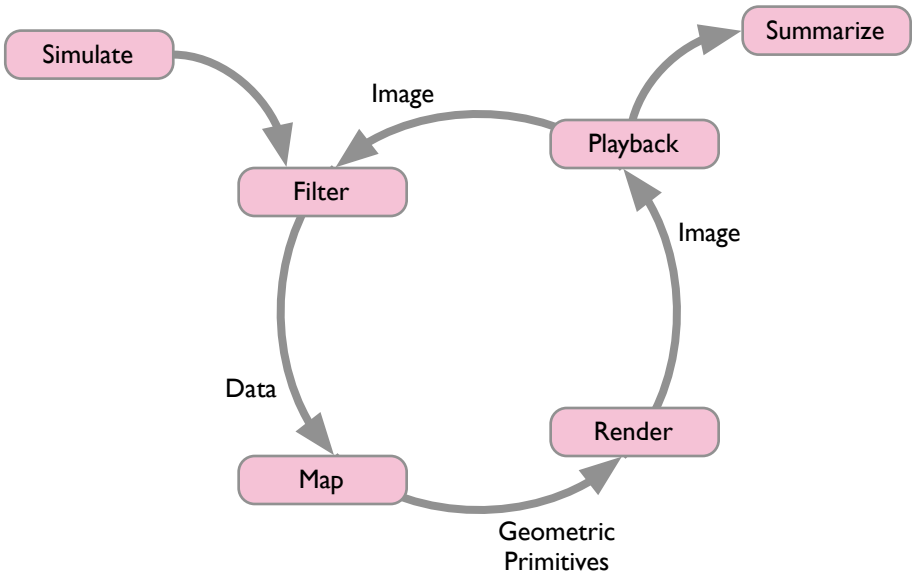


Figure 2.2: AVS's analysis cycle, the first known formalized visualization process model [reproduced from Upson et al., 1989, p. 32].

ConMan illuminates the visualization process through its deconstruction of the process into discretized logical steps that are visually represented within the user interface. In doing so, it empowers users with the insight of how a visualization was created, thereby facilitating the analyst's interpretation of the visualized data with this understanding.

2.3.2 Application Visualization System's Visualization Cycle

Continuing with the data flow metaphor is work by Upson et al. [1989] on the AVS, a visualization environment designed for visualizing scientific and engineering data. ConMan's modular component design, organized as a directed acyclic graph in a visual programming environment, was refined in the AVS, where individual components were classified by their functions. This classification comes from Upson et al.'s modelling of the visualization process as an unending cycle that is only completed once all analysis questions have been resolved. Figure 2.2 illustrates this analysis cycle, composed of four major functional classifications: filter, map, render, and playback. These classifications are defined as follows:

Filtering: involves processing the incoming raw data from a simulation or feedback from a previ-

ously rendered image into a form that is readily consumable by subsequent operations in the visualization process. It is characterized as: "Filtering operations include derived quantities such as the gradient of an input scalar field, integrative processes (for example, deriving flow lines from a velocity field), or simply extracting a portion of the solution".

Mapping: transforms data processed in the filtering stage into geometric primitives. There are numerous visualization design decisions on how this mapping can occur. To demonstrate this point, Upson et al. illustrate the many different options available for mapping a 3D scalar field to geometric primitives, shown in Figure 2.3.

Rendering: generates an image from the geometric model. Again, numerous options are available concerning the viewing transformation, projection transformation, clipping, colour, texture, lighting, and shading parameters used in the rendering process.

Playback: is the assessment of the rendered visualization by the researcher or engineer engaged in the analysis cycle. Examination of the visualization can lead to new analysis questions, which starts the analysis cycle all over again. This process continues until all analysis questions about the physical mechanism under study have been addressed.

ConMan provided a layer of abstraction from the details of the visualization process through the use of the data flow metaphor and the discretization of steps along the pipeline. AVS helped in formalizing this layer of abstraction by functionally classifying these discrete steps, providing a new mental model for conceptualizing the visualization process. Furthermore, this process model also served as the underlying system architecture of the AVS environment. To the best of my knowledge, it is the first known formal characterization of a visualization process model presented in the literature.

2.3.3 Haber and McNabb's Data Flow Model

The term "data flow model" was first coined by Haber and McNabb [1990] as a conceptual model for visualization workflow [Duke et al., 2006]. The model, shown in Figure 2.4(a), refines earlier work by Upson et al. [1989] in the way of terminology and expands the theoretical characterization of the visualization process. Haber and McNabb modelled the visualization process as a series of

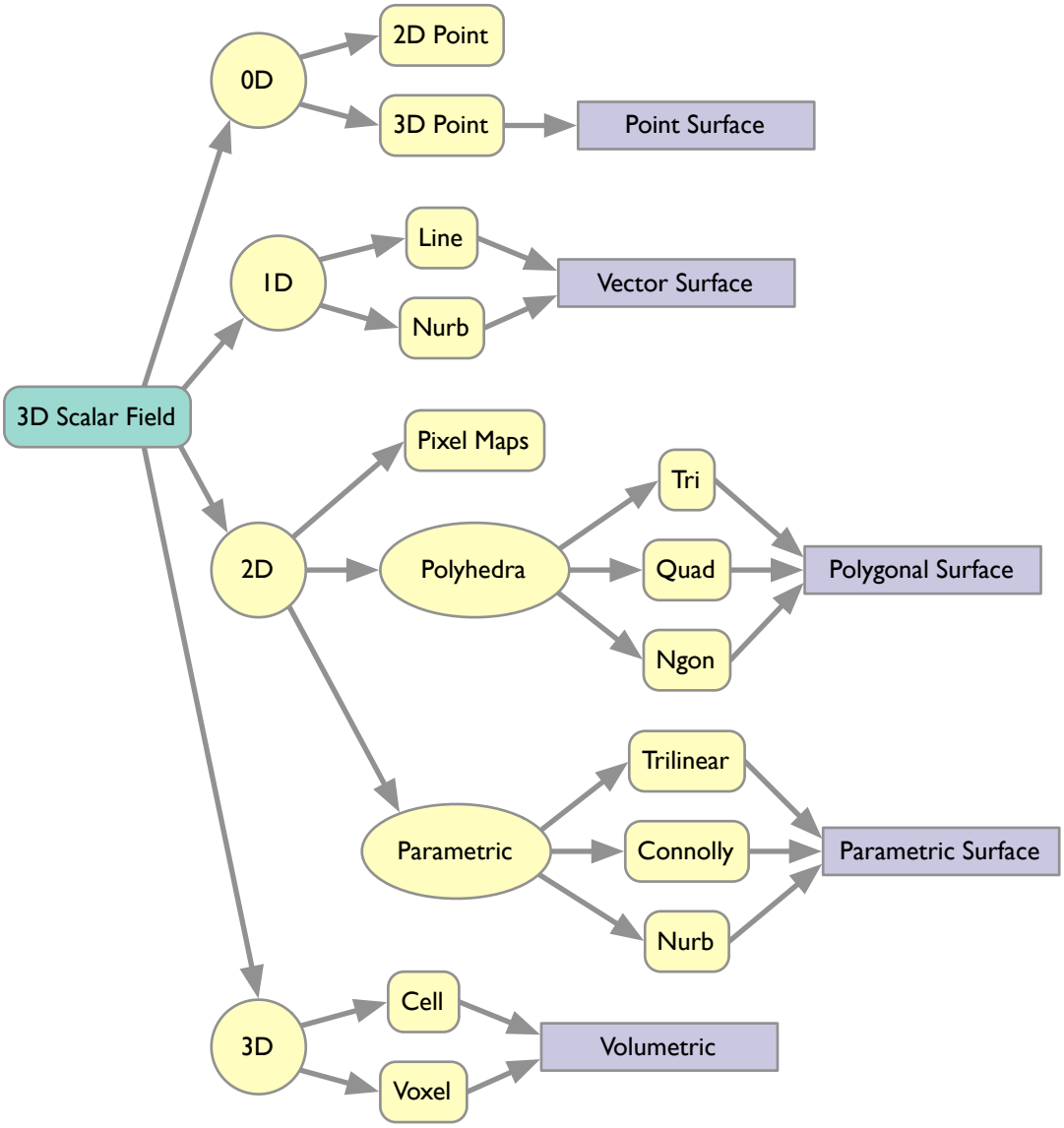


Figure 2.3: The numerous visualization design decisions available for mapping a 3D scalar field to geometric primitives, available in the mapping stage of AVS's visualization cycle [redrawn from Upson et al., 1989, p. 33].

three major transformations: data enrichment/enhancement, visualization mapping, and rendering. Like AVS's model, these major transformations can be composed of multiple operations which are grouped by their functional affordances, and are defined as follows:

Data Enrichment/Enhancement: is closely akin to AVS's "filtering" transformation where raw data from a simulation is processed into a form that is more coherent and accessible by subsequent

operations in the visualization process. The product of this transformation is identified as *derived data*, as the raw simulation data is fundamentally altered in one or more ways. Example operations include extracting a subset from the simulation data (filtering), interpolation, noise reduction, and deriving surface normals from a curve, surface, or hypersurface.

Visualization Mapping: involves transforming derived data into imaginary objects, referred to as Abstract Visualization Objects (AVOs). An AVO is described through a series of attributes, which "might include geometry, time, colour transparency, luminosity, reflectance, and surface texture" [Haber and McNabb, 1990]. This transformation combines AVS's entire "mapping" and parts of the "rendering" transformations. In Haber and McNabb's model, operations like assigning colours and textures to objects are classified as "visualization mapping" transformations, not rendering.

Rendering: generates an image from the AVO. Similar to the AVS transformation by the same name, with the exception of operations that have been moved into the visualization mapping transformation. Typical operations include viewing transformation, projection transformation, clipping, rasterization, *etc.*.

Haber and McNabb [1990]'s data flow model present refined definitions of the concepts first introduced by in earlier work by Upson et al. [1989]. The changes are subtle rather than profound, though the work has been credited as establishing a vocabulary of concepts for understanding the process of visualization [Duke et al., 2004]. One criticism of the data flow model is the poor precision in the classification of operations. The model is ambiguous, as it lacks a formal definition of the three major transformations and the borders that separate them. For example, consider a transfer function for deriving flow lines from a simulated velocity field. Based on the transformation definitions, such an operation could be classified as either a data enrichment/enhancement or a visualization mapping transformation. The flow lines operation derives new data from the initial simulation data (data enrichment/enhancement transformation), however the operation could also be viewed as mapping attributes into AVOs (visualization mapping transformation). Hence the vague demarcation of transformation classifications.

2.3.4 Use and Extension of the Data Flow Model

The data flow model has received widespread use in the SciVis community [Chi, 2002; dos Santos and Brodlie, 2004]. Haber and McNabb's original definition of the model is not rigidly adhered to, but rather the general abstraction is utilized: conceptualizing the visualization process as the flow of data through a pipeline of data transformations. This pipeline is a directed acyclic graph, where vertices represent data transformations and edges are the flow of data between transformations. In classifying these transformations it is not uncommon to see a mixing of the terminology used by Upson et al. [1989] and Haber and McNabb [1990], as is the case in Roberts [1998] and Brodlie [2005]. A notable extension of the data flow model is by dos Santos and Brodlie [2004] who added support for working with multidimensional and multivariate data.

2.4 Data State Model

The data state model emerged out of the InfoVis community as a way of providing further precision in the description of the visualization process, as well as unifying its interaction model [Chi and Riedl, 1998]. The pipeline metaphor is again used in this model with particular attention being paid to the changes in the data's state at each step in the pipeline [Jankun-Kelly, 2003], taking a state-based perspective of the visualization process. The model was first introduced by Chi and Riedl [1998], with later refinements by Card et al. [1999] and Carpendale [1999].

Chi and Riedl's *operator interaction framework* introduced innovative contributions to the conceptualization of the visualization process. The research which I present in this thesis relies heavily on this framework and the important theoretical considerations which it identifies. Therefore, I pay careful attention to these issues discussing them in detail.

2.4.1 Operator Interaction Framework

The *operator interaction framework* proposed by Chi and Riedl [1998] was the first to formalize the concept of looking at the visualization process from the perspective of a data state model. This framework builds on earlier work by Card and Mackinlay [1997] on describing the design space of information visualization. Like the data flow model, it too models the visualization process as a directed

acyclic graph. Where the two models begin to differ is in the meaning of the vertices and edges of the graph: in the data state model, vertices represent data states and edges are the transformations between data state; here, vertices provide information as to the status of the data within the pipeline. Recall, that in the data flow model, vertices represent data transformations and edges are the flow of data between transformations. Despite these differences, the data flow and the data state models have been shown to be equivalent in visualization expressiveness [Chi, 2002]; that is, the same visualization can be represented using either model.

2.4.1.1 Chi and Riedl's Visualization Pipeline

Chi and Riedl's proposed visualization pipeline is shown in Figure 2.4(b), and is closely akin to Haber and McNabb's version. The pipeline begins with *data* in its raw form. From this state, a *data transformation* processes the data into the *analytical abstraction* state, deriving "metadata, data about data, or information" [Chi and Riedl, 1998] from the raw data. The analytical abstraction state is described as "processed data that is not yet mappable but include[s] all information from the raw data that will be visualized" [Chi, 2002]. From the analytical abstraction state, a *visualization transformation* is applied, resulting in the *visualization abstraction* state. This transformation applies a visual form to the information in the analytical abstraction state; multidimensional scaling and clustering are examples of possible operations [Chi and Riedl, 1998]. In the visualization abstraction state, information "is mappable and visualizable on the screen using at least one visualization technique" [Chi, 2002]. From the visual abstraction state, a *visual mapping transformation* is applied resulting in the end view of the visualized data. Notice that the these three transformations—data, visualization, and visual mapping—closely parallel the respective transformations—data enrichment/enhancement, visual mapping, and rendering—in Haber and McNabb's model.

A novel contribution from Chi and Riedl's pipeline is the attention drawn to the *data space*, or *system control*, and the *view space*, or *analyst control*. These two spaces are seen as end points on a spectrum, a *visualization pipeline scale* [Chi and Riedl, 1998], indicating the amount of direct manipulation available with the different pipeline processes. At the view state, the analyst using the visualization system can interact heavily with the processes involved in the visual mapping transfor-

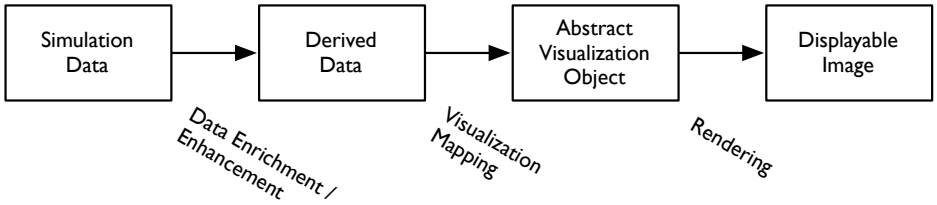
mation (view space). For example, rotating a view or applying a different colour scale can all be done with ease. Moving away from the view state towards the data stage, the amount of direct interaction decreases, as the system plays a more dominate role and the analyst's interactions become more indirect (data space). Processes like parsing and loading in a file, are handled by the system, and the analyst's involvement is minimal.

2.4.1.2 Classification of Operators in the Operator Interaction Framework

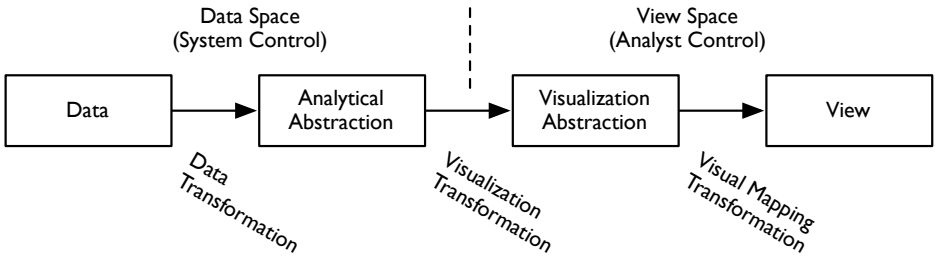
The state-based visualization pipeline is an important part of Chi and Riedl's *operator interaction framework*. One of the affordances provided by this framework is in the classification of operators. In this context, an *operator* is defined as any interaction of the analyst with the system, be it directly or indirectly. In creating an ontology, the framework builds off of two observations about the fundamental properties of operators: operational versus functional similarity, and value versus view orientation.

The first observation suggests that operators can be classified based on whether or not their technical implementation is operationally or functionally similar. *Operational similarity* is when the implementation of an operator is, for all intensive purposes, identical across applications regardless of the domain. An example of operational similarity is an affine transformation used in computer graphics, such as translation, rotation, and scaling. Applications ranging from medical volumetric visualization or text visualization all utilize the same matrix operations for translation, rotation, and scaling when handling graphics primitives. *Functional similarity* is when the semantics of an operator is equivalent across application domains, however its implementation is significantly different. Filtering is an example that is conceptually universal, however the technical implementation for filtering three dimensional structures in a volumetric data set versus words in a text corpus, are radically different.

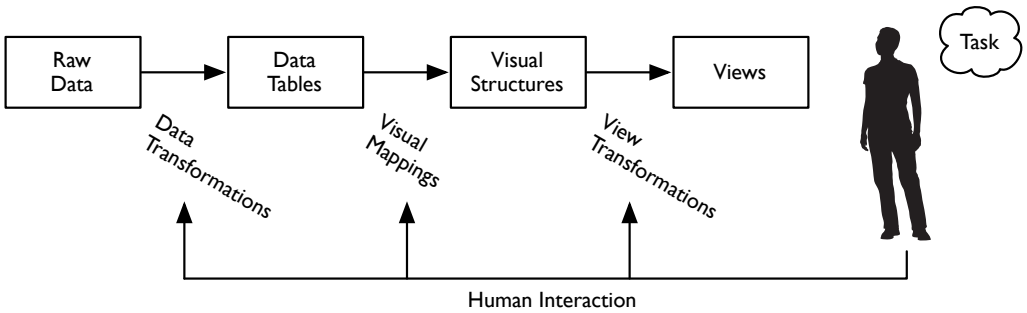
The second observation suggests another classification of operators based on the operator's value or view orientation. A *value* operator makes a fundamental change to the underlying data source, creating an entirely new data set. Clustering an unstructured data set into a hierarchy is an example of a value operator, as the product of the clustering operation is a fundamentally new data set. A



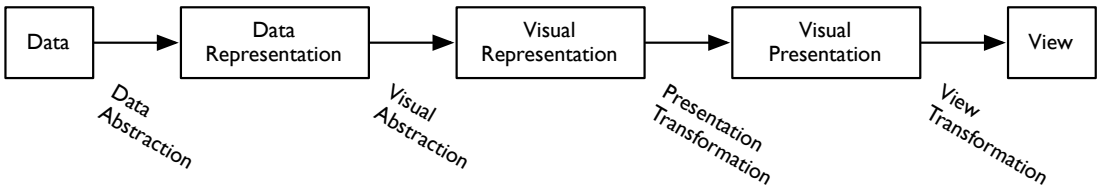
(a) The visualization pipeline according to the *data flow model* proposed by Haber and McNabb [1990], which refined earlier work by Upson et al. [1989] on the AVS visualization cycle (see Figure 2.2).



(b) The visualization pipeline according to the *data state model* proposed by Chi and Riedl [1998]. It builds off earlier work by Haber and McNabb [1990], taking a state-based perspective of the visualization process and adding the control space separation.



(c) The *visualization reference model* proposed by Card et al. [1999], which refined Chi and Riedl [1998]'s pipeline by modifying the terminology and removing the control space separation, making each state under the analyst's control.



(d) The visualization pipeline used by Carpendale [1999], based on Chi and Riedl [1998]. While conceptually similar to its progenitor, Carpendale uses different terminology and adds a *presentation space* to the pipeline.

Figure 2.4: Visualization process models which have evolved incrementally, building on earlier work with subtle refinements to process terminology and characterization.

view operator makes a superficial change to the visualization without fundamentally altering the underlying data set which it represents. Object translation, rotation, and scaling are examples of view operations, as they make peripheral changes to the representation of the underlying data set behind the object.

These classification types—operational versus functional, value versus view—are in fact related to one another. Value operators, which fundamentally alter the underlying data set, are typically functionally similar across application domains. View operators, which affect the presentation of a visualization, are typically operationally similar. Returning to the previous example, a value operator such as clustering unstructured data is conceptually similar across applications, however its implementation is strongly dependent on the data type, and therefore is operationally similar. On the other hand, a view operator such as affine transformation for translation, rotation, and scaling of graphics primitives is functional similar. Moreover, the dichotomous relationship between operational similarity/value orientation and functional similarity/view orientation are end points on spectrum, rather than binary in nature. This spectrum integrates into the visualization pipeline, beginning with raw data, which is characterized by operators that are operationally similar and value orientation, and ending in a view, with operators that are functionally similar and view orientated. Operators can be positioned within the framework, as illustrated by Chi and Riedl's original figure shown in Figure 2.5.

2.4.1.3 Value Versus View Operators

One major benefit of the operator interaction framework is the powerful mental model which it provides and that can be applied to understand the mechanics of the visualization process. To illustrate the model's potential, consider the following example.

Figure 2.6(a) shows a simple pie chart visualization with four equally weighted data items. Perhaps we are only interested in items A and D, and would like all other items filtered out. There are two possible ways of realizing this operation: filtering data items and then visualizing the resulting subset, or visualizing and then filtering. The results of these two possibilities are shown in Figure 2.6(b) and Figure 2.6(c). What type of filtering did we want to do? These different interpretations of intentions

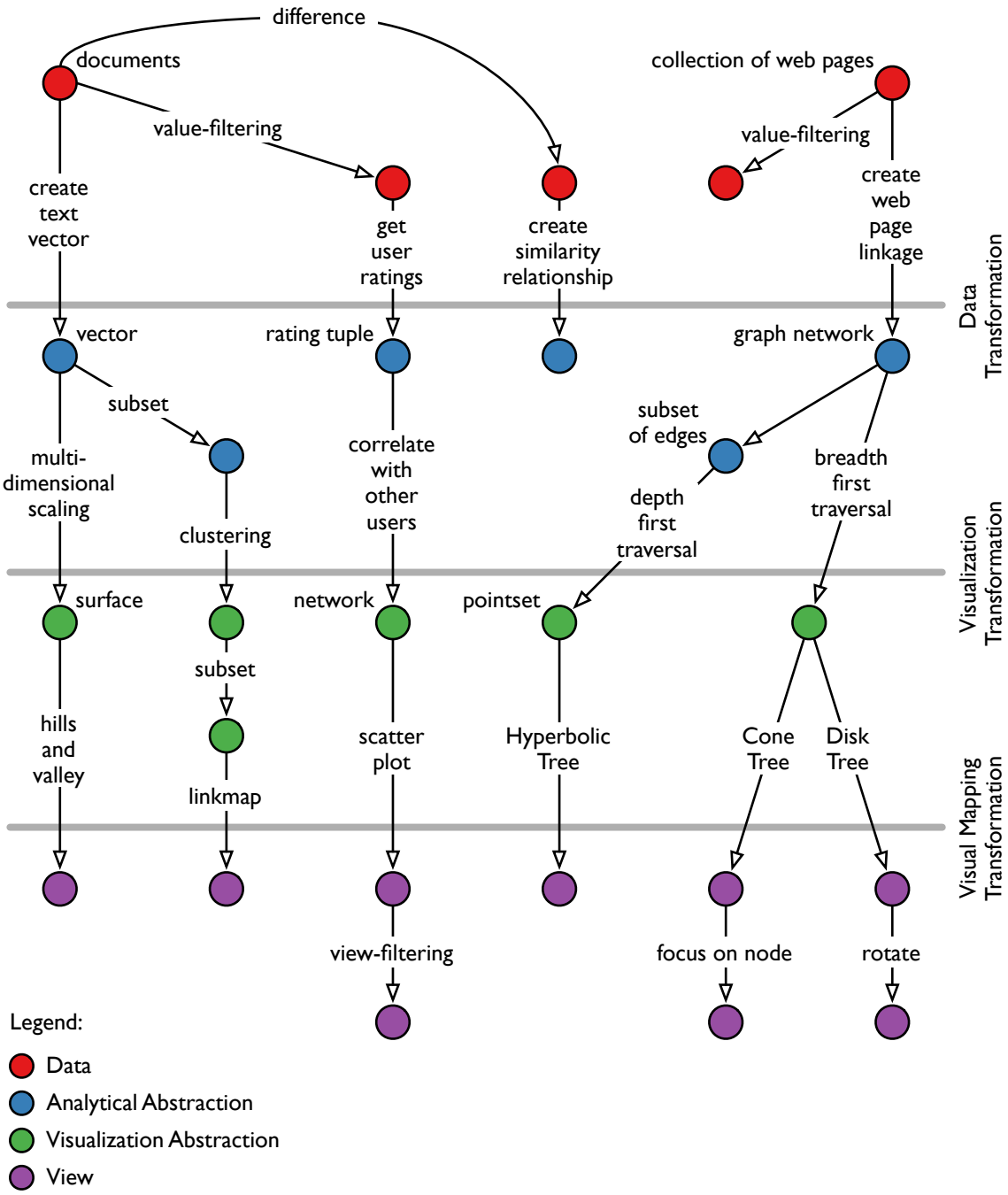
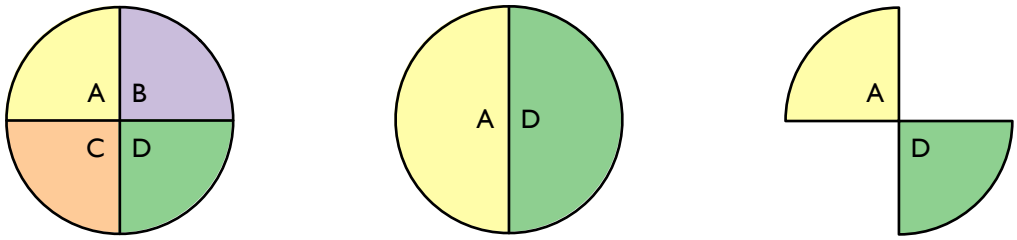


Figure 2.5: An example of operator classification within the operator interaction framework [adapted from Chi and Riedl, 1998, p. 66].



(a) The original pie chart visualization without any filtering of data items.

(b) An example of a value operation: data items B and C are filtered from the data set and the resulting subset is then visualized.

(c) An example of a view operation: the complete data set if first visualized, and then data items B and C are filtered from the view, keeping the spatial organization of the non-filtered items.

Figure 2.6: An example of the distinction between value versus view operators using a filtering operation.

leads to what Chi and Riedl identify as the *gulf of execution*, defined as the "difference between the intentions and the allowable possible actions" [Norman, 1990, chap. 2]. By using the operator interaction framework, we can understand how these two types of filtering operations differ, thereby bridging the gulf of execution by matching our intentions with unambiguous system actions.

The difference between these two filtering operations is that one of them is a value operator, and the other is a view operator. Filtering the complete data set into a subset is a value operator. The operator is then followed by the visualization transformation and the end view shows no signs of the removed data items (Figure 2.6(b)). Whereas filtering after the visualization transformation has been applied is a view operator. It simply removes those visualized data items from the end view, while keeping the spatial organization of the non-filtered items (Figure 2.6(c)). Both types of filtering are valid and useful in the right situation; the issue is that the analyst using the visualization system must be aware of these differences. The operator interaction framework provides a mental model to readily make sense of these differences.

2.4.2 Visualization Reference Model

The operator interaction framework and its visualization pipeline were later refined by Card et al. [1999]. The refinement sought to provide a simple reference model to facilitate the discussion of information visualization systems and aid in their comparison [Card et al., 1999, p. 17]. Card et al.

termed their visualization pipeline the *visualization reference model*, shown in Figure 2.4(c).

A series of transformations modelled as a directed *cyclic* graph, which maps raw data into an end visualization. The process is described as follows:

> *Data Transformations* map *Raw Data*, that is, data in some idiosyncratic format, into *Data Tables*, relational descriptions of data extended to include metadata. *Visual Mappings* transform Data Tables into *Visual Structures*, structures that combine spatial substrates, marks, and graphical properties. Finally, *View Transformations* create *Views* of the Visual Structures by specifying graphical parameters such as position, scaling, and clipping. User interaction controls parameters of these transformations, restricting the view to certain data ranges, for example, or changing the nature of the transformation. The visualizations and their controls are used in service of some task.
>
> The core of the reference model is the mapping of a Data Table to a Visual Structure. Data Tables are based on mathematical relations; Visual Structures are based on graphical properties effectively processed by human vision. Although Raw Data can be visualized directly, Data Tables are an important intermediate step when the data are abstract, without a direct spatial component. [Card et al., 1999, p. 17]

The visualization reference model simplifies Chi and Riedl's previous work and removes the control space separation, making each state under the analyst's control [Collins, 2010]. The model lacks the rigour of the operator interaction framework, however its original goal was for a general purpose, simple model for discussing information visualization systems. To this end the model has been very successful. It is commonly used in the InfoVis literature and in the design of information visualization systems (e. g., prefuse [Heer et al., 2005]).

2.4.3 Carpendale's Presentation Space

A notable extension of Chi and Riedl's pipeline is work by Carpendale [1999], modifying the terminology used in the pipeline and adding a *presentation space* to pipeline. Carpendale's version of the visualization pipeline is shown in Figure 2.4(d). Here a distinction is made between the *representation* and the *presentation* of information in the following manner:

> Representation is the act of creating a basic image that corresponds to the information such as creating a drawing of a graph. Presentation is the act of displaying this image, emphasizing and organizing areas of interest. For example, a map of Vancouver may be presented with one's route to work magnified to reveal street names. [Carpendale, 1999, p. V]

Therefore, the visualization abstraction state in Chi and Riedl's pipeline is divided into two unique states: visual representation and visual presentation. I adopt this division in the visualization model used in my thesis research.

2.4.4 Visualization Pipelines Compared

On a cursory glance the various visualization pipelines appear very similar to one another. With subtle refinements in terminology for pipeline stages and transformations, how are the different pipeline distinguished from one another? To illuminate this question, let us classify the individual operations in an example visualization process according to each of the visualization pipelines. The top of Figure 2.7 presents one such example series of visualization operations, and below it, the classification of each operation within the four visualization pipelines from Figure 2.4: the data flow model [Haber and McNabb, 1990], the data state model [Chi and Riedl, 1998], the visualization reference model [Card et al., 1999], and the visualization pipeline from Carpendale [1999]. As shown in Figure 2.7, the four process models achieve consensus on three points: the range of operations included in the initial *data* stage, the distinct edge in each model for the transformation of *data abstraction* into the *representation* stage through the application of a *node-link layout* transform, and the termination of the whole process with the *view* stage. (Albeit, the visualization reference model includes a feedback loop between the visualization pipeline and the individual engaged with the visualization and his/her driving task, which has not been shown in this comparative figure.) Figure 2.7 shows the different levels of resolution in each of the pipelines—for instance the data state model from Chi and Riedl [1998] classify the final three operations of the example visualization process as occurring in their definition of the view state, whereas Carpendale [1999] defines distinct states for each of these operations. Moreover, while the various visualization process models appear initially

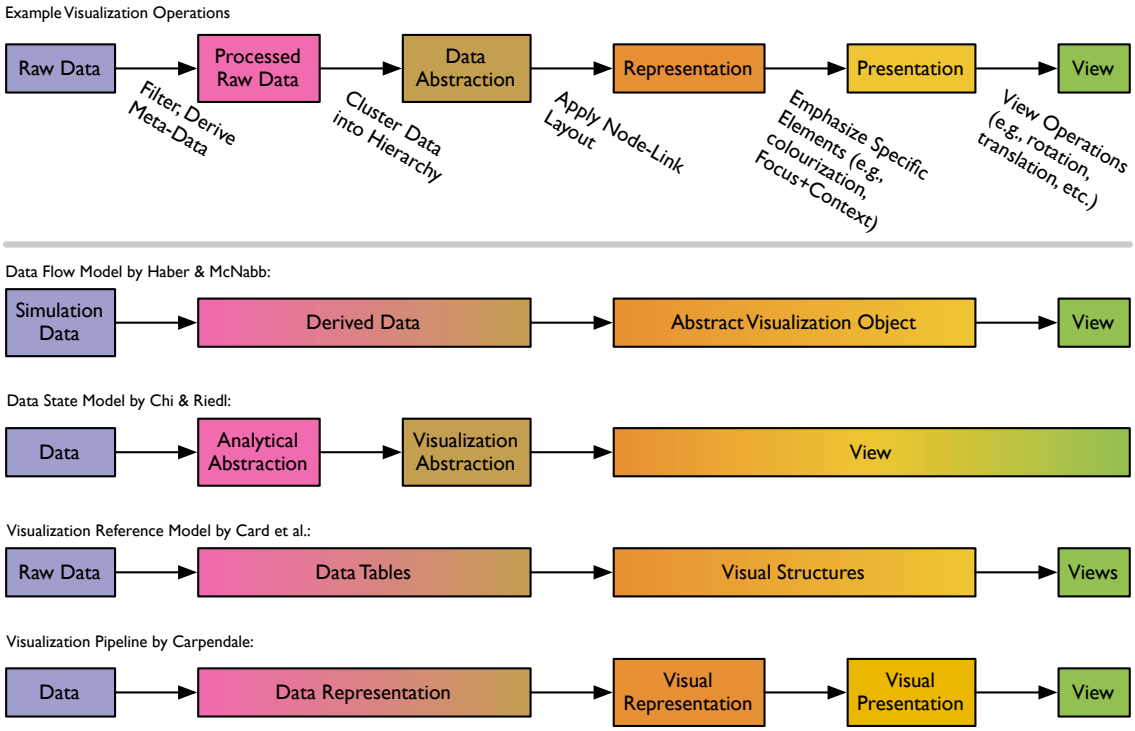
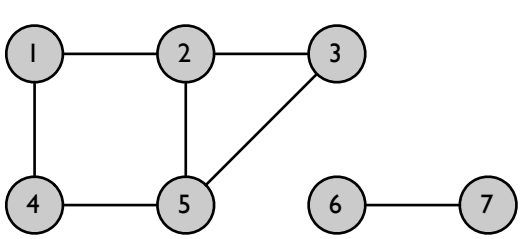


Figure 2.7: Categorization of individual operations from an example visualization process within the different visualization pipelines. The individual operations in the example visualization process (shown at the top) are qualitatively colourized; this colour is assigned to the stage in the various visualization pipelines that the operation is classified under.

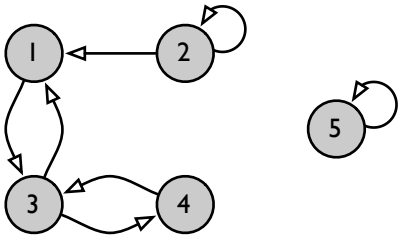
similar, the classification of operations can vary between the different models.

2.5 Hierarchical Data

Thus far, our discussion has focused on the theoretical considerations of the visualization process. In moving to more pragmatic territory, I now turn my attention towards the visualization of a particular type of data: hierarchical data. In this section, I introduce a formal definition of hierarchical data and present a few visualization techniques for visualizing this type of data. The Lark system uses hierarchical data as an illustrative example to present the system's concepts for supporting collaboration. Therefore, visualization techniques for hierarchical data are not a contribution of this work.



(a) An undirected graph $G = (V,E)$, where $V = \{1,2,3,4,5,6,7\}$ and $E = \{(1,2),(1,4),(2,3),(2,5),(3,5),(4,5),(6,7)\}$.



(b) A directed graph $G = (V,E)$, where $V = \{1,2,3,4,5\}$ and $E = \{(1,3),(2,1),(2,2),(3,1),(3,4),(4,3),(5,5)\}$.

Figure 2.8: Examples of undirected and directed graphs, illustrated using node-link diagrams.

2.5.1 Definition of a Tree

To introduce what hierarchical data is, first consider the following example. Before you now is a thesis. Like many other documents, this thesis is organized into chapters. Each individual chapter is broken into sections and some of these sections are further divided into subsections, like the one you are reading now: Section 2.5.1. The structure I have just described is an example of hierarchical data. Individual elements—such as documents, chapters, sections, and subsections—are organized relative to one another using a parent-child relationship. The thesis document (parent) is made up of chapters (children); a chapter (parent) is made up of sections (children), and so on. These relationships can be represented using a hierarchical tree structure, modelled mathematically as a *connected, acyclic, undirected graph*, also known as a *rooted tree* [Cormen et al., 1990, p. 93].

2.5.1.1 Graph

In graph theory, a *graph* $G = (V,E)$ is a mathematical structure [Gross and Yellen, 2006, p. 2] composed of a pair (V,E) where V is a finite, non-empty set and E is a set of pairs on V [Cormen et al., 1990, p. 86]. The set V is composed of elements called *vertices* (also referred to as *nodes*) and the set E is composed of pairs of vertices called *edges*. Graphs can either be undirected or directed.

In an *undirected* graph $G = (V,E)$, the set E are unordered pairs of edges, such that edge $e = \{u,v\}$ where $e \in E$ and $u,v \in V$. Since edges do not have a direction the following two edges are equivalent: $\{u,v\} = \{v,u\}$. Self loops are not permitted in undirected graphs, therefore $u \neq v$. Figure 2.8(a) illustrates an example of an undirected graph.

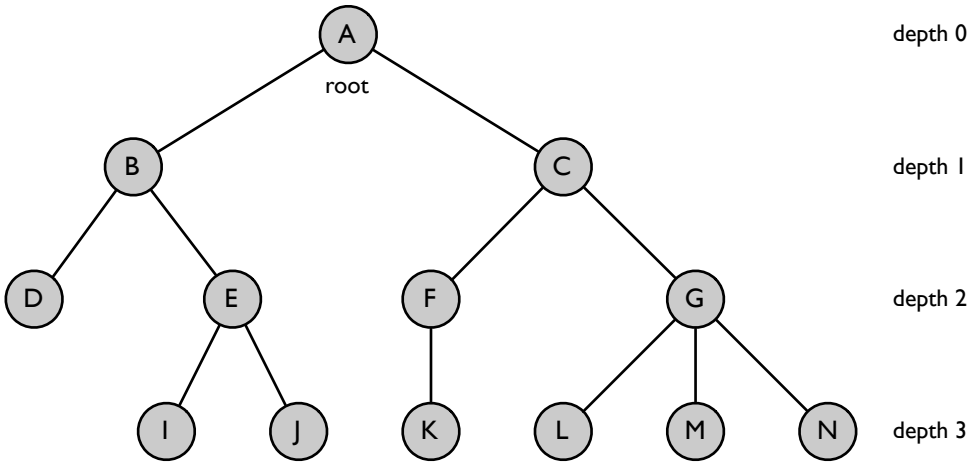


Figure 2.9: An example of a rooted tree, illustrated using a node-link diagram. The *root* of the tree is vertex letter A and tree contains seven *leaf* vertices: D, I, J, K, L, M, and N. Vertex letter C is the *parent* of vertices F and G, which are *siblings* of one another and children of vertex C. Vertex letter G has a *degree* of three and is at a depth of two.

In a *directed* graph $G = (V, E)$, the set E are ordered pairs such that for each edge $e = \{u, v\}$ where $e \in E$ and $u, v \in V$. This is similar to a undirected graph, with two exceptions: self loops are allowed, $u = v$, and edges are order pairs of vertices, $\{u, v\} \neq \{v, u\}$. Figure 2.8(b) illustrates an example of a directed graph.

Within a graph $G = (V, E)$, a *path* is a sequence of vertices connecting vertex u and u'. This sequence of vertices, denoted $\langle v_0, v_1, v_2, \ldots, v_k \rangle$, connects vertices u and u' to one another such that $v_0 = u$, $v_k = u'$, and $\{v_i, v_{i-1}\} \in E$ for $i = 1, 2, \ldots, k$. The *length* of a path is k. If all vertices in the path are distinct, the path is considered *simple* [Cormen et al., 1990, p. 87]. A path is considered a *cycle* if $v_0 = v_k$ [Cormen et al., 1990, p. 88]. An *acyclic* graph is a graph that has no cycles.

In a *connected* graph $G = (V, E)$, for each pair of vertices $\forall \{u, u'\} \in V$ where $u \neq u'$, there exists some path connecting u to u' [Gross and Yellen, 2006, p. 32].

2.5.1.2 Rooted Tree

A *rooted tree* is a connected, acyclic, undirected graph with one distinguished vertex known as the *root* of the tree [Cormen et al., 1990, p. 91]. In a rooted tree, or simply a *tree*, $T = (V, E)$ with root vertex r, for any non-root vertex u in the tree, there is a unique path from r to u. Along this path, for the last edge $\{y, u\}$, we say that y is the *parent* of x, and x is the *child* of y. Each vertex in the tree has

a parent vertex, with the exception of root. *Siblings* are any two vertices with the same parent. *Leaves* are vertices that do not have any children. The *degree* of a vertex is the number of children it has. The *depth* of a vertex x is the length of the unique path from the root r to x (this can also be referred to as the *level* of a vertex) . An example of a rooted tree is shown in Figure 2.9.

2.5.2 Visualization of a Tree

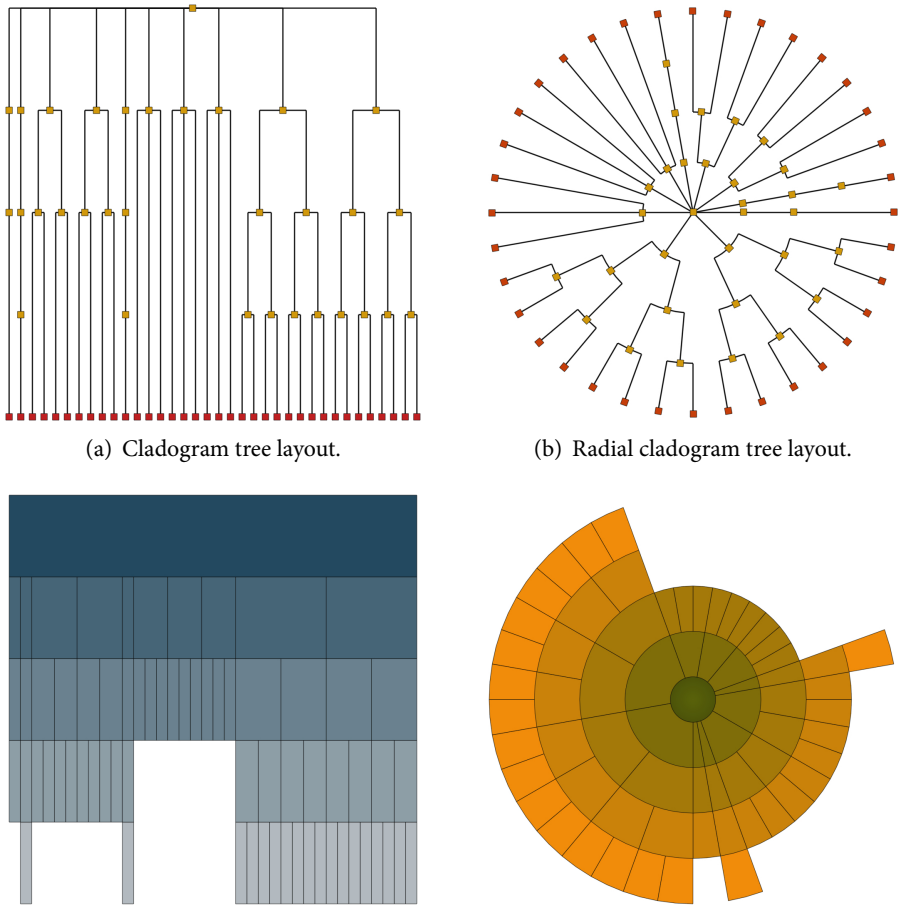
Numerous visual representations of rooted trees have been created, and they are commonly referred to as *tree layouts*. In this section, I present an overview of a few of these different types of two-dimensional layouts, focusing on the conventional variants and the types used in the Lark system. Notable tree layouts, which are not discussed, include treemaps [Shneiderman, 1991] and layouts for non-two-dimensional space, such as those for hyperbolic space [Munzner, 1997].

2.5.2.1 Node-Link Representation

By far the most commonly used method for visually representing a tree is a *node-link* representation. Here, the tree vertices are represented using shapes, such as circles or squares, and edges are represented using connecting lines. There are numerous variants of node-link layouts, however, this discussion is limited to rectilinear, cladogram, and radial cladogram layouts.

Rectilinear Layout: An example of a rectilinear node-link tree layout is shown previously in Figure 2.9. Assuming that we are drawing the tree vertically on a plane in a top-down fashion, the spatial arrangement of vertices places the root vertex at the top, with the children of the root horizontally aligned directly bellow their parent (the root). This layout structure is repeated recursively on the root vertex's children, until all vertices in the tree have been positioned. To make room for all the leaf vertices in the tree, vertices may be horizontally repositioned to avoid cluttered and potentially overlapping edge lines. With this layout algorithm, all vertices of the same depth are horizontally aligned with one another, as seen in Figure 2.9. Developing fast and efficient node-link layout algorithms that adhere to certain aesthetic requirements continues to be an active area of research [Buchheim et al., 2002; Marriott and Sbarski, 2007].

Cladogram Layout: A *cladogram* (also known as a *dendrogram*), is often used as a general term for



(a) Cladogram tree layout.

(b) Radial cladogram tree layout.

(c) Icicle plot tree layout.

(d) Sunburst tree layout.

Figure 2.10: Four example tree layouts, all visualizing the same data set with different representations.

a tree diagram [Baum and Offner, 2008] that is commonly used in illustrating hierarchical data which represents how related individual items in a set are to one another. Cladograms are widely used in evolutionary biology for depicting the phylogenetic relationships between species, and in cluster analysis applications for illustrating the relatedness between clustered elements. An example of a cladogram layout is shown in Figure 2.10(a). A cladogram is very similar to the rectilinear node-link layout, with the notable distinctions that all leaf vertices are aligned along the bottom of the tree, regardless of their depth, and that the edge lines are drawn with 90° angles.

Radial Cladogram Layout: A *radial cladogram* is similar to a regular cladogram, except that it is laid

out circularly rather than rectilinearly, as shown in Figure 2.10(b). The radial layout makes better use of the available space, partitioning leaf vertices along the circumference of the outer ring of the tree. This affordance makes the radial cladogram a common choice for illustrating hierarchical data when space is at a premium; for example, in publication figures (e. g., [Willis et al., 2008]).

2.5.2.2 Space Filling Representation

The three tree layouts we have discussed so far—node-link, cladogram, and radial cladogram—are typically categorized as node-link representations. Another classification of tree layouts are those that are *space-filling*, such as the *icicle plot* and *sunburst* layouts.

Icicle Plot Layout: Figure 2.10(c) illustrates an example of a *icicle plot* layout. Vertices in the icicle plot are positioned in a similar fashion to the node-link layout (see Section 2.5.2.1), except that vertices are drawn to fill the area of the level on the tree that they are on. Edges are implicitly represented, as a vertex's children cascade downward from their parent.

Sunburst Layout: A *sunburst* layout is a radial, space-filling layout similar to the icicle layout, except for the radial arrangement of vertices. The sunburst layout is to the icicle layout, as the radial cladogram is to the rectilinear cladogram. This layout was introduced by Andrews and Heidegger [1998], and is shown in Figure 2.10(d).

2.6 Collaboration

Moving from individual visualization techniques, in this section I review existing research on facilitating collaborative work practices in digital environments. As introduced in Section 1.2, my thesis research investigates collaborative work practices that make use of information visualizations during data analysis tasks. In particular, I am interested in collaborative teamwork that is occurring in a synchronous co-located environment, with small groups of people who are using a large multi-touch tabletop display. My investigation of collaborative teamwork is centred around supporting mixed-focus collaboration, and the following discussion is focused by this research scope.

I begin our discussion with work from the Computer-Supported Cooperative Work community on mixed-focus collaboration. This is followed by research on collaboration around interactive tabletop displays and an overview of design guidelines for supporting effective co-located collaboration around tabletop display. Next, I present an overview of InfoVis research on synchronous co-located collaborative work. I conclude our collaboration discussion with a synopsis of design guidelines for co-located collaborative information visualization systems.

2.6.1 Computer-Supported Cooperative Work

Work in Computer-Supported Cooperative Work (CSCW) has indicated that the discrete activities which occur during collaborative group work sessions, such as brainstorming or planning, cannot be strictly dichotomized as either *independent* or *shared* activities [Tang et al., 2006]. The distinction is rather a spectrum, which collaborative activities fluidly and frequently move within. The frequently transition of group members between loosely coupled, individual work and tightly coupled, group work is the defining feature of *mixed-focus collaboration* [Gutwin and Greenberg, 1998]. In this context, coupling is "the degree to which people are working together" [Baker et al., 2002]. A strong theme within research into collaboration is how to support different types of individual and group work, as well as the transitions between these activities. The frequency of changes in collaborative coupling means that these transitions should be quick and easy to perform [Elwart-Keys et al., 1990; Mandviwalla and Olfman, 1994].

In designing computer interfaces which support mixed-focus collaboration, designers are often confronted with the trade-off of designing for the individual or the group at large [Gutwin and Greenberg, 1998]. In Tang et al. [2006], mixed-focus collaboration was investigated through two observational studies which looked at the influence of viewing techniques on collaborative coupling. The viewing techniques used in these studies presented different affordances for individual and group work, ranging from a single shared data representation, to multiple independent representations. Tang et al. [2006] found that with a single shared representation, an individual's ability to work independently may be compromised, yet using separate copied views may prevent many group collaborative dynamics from emerging. It was therefore noted that a trade-off is necessary between supporting

the individual versus the group.

2.6.2 Tabletop

Since the early explorations of interactive tabletop displays, researchers have sought to leverage people's well established understanding of traditional tabletops in designing novel interfaces and interaction techniques for digital tabletops [Wellner, 1993; Krüger et al., 1995]. This approach is still found in recent tabletop research. Researchers using observational based methodologies begin with studying people's behaviour in traditional tabletop environments and then apply these observations in designing for the digital realm [Kruger et al., 2004; Scott et al., 2004; Isenberg et al., 2008]. Other researchers have taken different approaches to developing systems and interaction techniques for digital tabletops [Shen et al., 2002, 2004; Wu and Balakrishnan, 2003], which are often then evaluated with formal user studies [Ryall et al., 2004; Ringel et al., 2004].

Scott et al. [2003a] characterizes a series of system design guidelines for *supporting effective co-located collaboration around tabletop displays*. Best practices along with lessons learned from previous tabletop research is collected and succinctly summarized in eight design guidelines. These guidelines are as follows:

Support Interpersonal Interaction: Collaboration is organized and coordinated through the interpersonal interactions of group members. Technology should not interfere with these interactions, as they are fundamental to the collaborative process. Work by Gutwin and Greenberg characterize these interpersonal interactions in what they call the *mechanics of collaboration*, "the low level actions and interactions that must be carried out to complete a task in a shared manner" [Gutwin and Greenberg, 2000]. These mechanics include: explicit communication, consequential communication, coordination of action, planning, monitoring, assistance, and protection; and can be evaluated according to their effectiveness, efficiency, and satisfaction. Again, each mechanic does not necessarily require direct technological support, but at a minimum, they should not be interfered with.

Support Fluid Transitions Between Activities: Switching between activities should impose little to no overhead, allowing collaborators to focus on the task at hand, rather than activity manage-

ment. Mixed-focus collaboration is characterized by frequent transitions between activities, and therefore, any incurred transition overhead would repeatedly penalize task performance.

Support Transitions Between Personal and Group Work: Switching between varying degrees of collaborative coupling occurs frequently [Elwart-Keys et al., 1990; Mandviwalla and Olfman, 1994], therefore these transitions should be performed with ease. Recent work by Isenberg et al. [2008] has provided additional evidence that these transitions require the coordination of group members' activities, and therefore, these transitions should be facilitated by support from the collaborative environment.

Support Transitions Between Tabletop Collaboration and External Work: The workflow of collaborative activities is rarely isolated to just the tabletop environment. For example, a group planning meeting might build off of initial ideas sketched out by individual group members prior to the collaborative session; or, the ideas generated in a group brainstorming session might be further developed afterwards by a single individual with a vested interest in the project. These examples illustrate that collaborative activities are often a stage within a larger workflow involving individual activities which take place beyond the collaborative tabletop environment. Harnessing the collaborative effort relies on integrating into this workflow and systems should support this.

Support the Use of Physical Objects: The horizontal surface provided by tables presents a flexible workspace which people can gather around and place objects upon. These objects might be relevant to the collaborative task, for example, project documents, or they may be unrelated, such as coffee mugs and personal day-timers. The familiar usage of tables should not be lost in the transition from traditional to digital tabletops. Rather, these well established practices should be augmented with digital features, providing a "seamless integration of digital and physical objects at the table" [Scott et al., 2003a].

Provide Shared Access to Physical and Digital Objects: Tables are particularly well suited for groups of people sharing information and objects with one another. The use of gestures and deictic references towards objects on the table, communicates rich spatial information to the

group [Bekker et al., 1995; Tang, 1991]. If the group is working with shared objects, a gesture towards an object has a direct spatial relationship, which promotes group awareness and focus [Suzuki and Kato, 1995]. When each group member has his/her own individual copy of an object, gestures have an indirect spatial relationship. Each group member must first identify the object being gestured towards by a colleague, locate his/her own copy of the particular object, and then spatially translate the gesture. This identification and spatial translation incurs cognitive overhead to important actions that were traditionally extremely lightweight [Bekker et al., 1995; Bly, 1988; Tang, 1991; Gutwin et al., 1996].

With groups of people gathering around a table, the orientation of objects in the workspace can both impede and facilitate group interaction [Tang, 1991]. Orientation-dependent objects, such as text documents, can be difficult to understand when facing away from the viewer [Wigdor et al., 2007]. At the same time, orientation has been shown to be "critical in how individuals comprehend information, how collaborators coordinate their actions, and how they mediate communication" [Kruger et al., 2003]. It is therefore important that digital tabletop technologies provide flexible, user-controlled spatial arrangement and orientation of objects in the workspace.

Consideration for the Appropriate Arrangements of Users: Similar to the flexible arrangement of objects on a table, technology should provide the same flexibility for people around the table. Different types of tasks around a table have different optimal spatial configurations that people feel most comfortable working in. For instance, in conversation based activities, a face-to-face arrangement is generally preferred by adults. On the other hand, "activities that require coordinated actions may best be supported by close user positions, because this positioning can enhance workspace awareness" [Scott et al., 2003a]. Furthermore, as group members transition between different degrees of collaborative coupling, their optimal spatial arrangement around the table may change as well. Technology must therefore support this, allowing collaborators to freely move about the tabletop.

Support Simultaneous User Actions: Concurrent interaction is a natural behaviour commonly ob-

served in groups of people working together around traditional tabletops [Tang, 1991; Scott et al., 2003b; Isenberg et al., 2008]. Digital tables must also provide support for concurrent interaction within the workspace from multiple group members. Recent advances in multi-touch technology (e. g., frustrated total internal reflection (FTIR) [Han, 2005]) have made this requirement feasible, empowering designers with the means of creating rich interaction techniques. With these hardware affordances, it is crucial that support is also provided at the software level. Tabletop systems should allow multiple tasks to be carried out simultaneously by multiple individual group members, thereby enabling synchronous collaboration to occur.

In summary, the design guidelines outlined by Scott et al. [2003a] mirror the ideas from early tabletop research [Wellner, 1993; Krüger et al., 1995] in that they emphasize the importance of leveraging people's well established understanding and experience with traditional tables when designing for digital tabletops. Technology should augment familiar modes of behaviour with computational resources and integrate into the large working environment.

2.6.3 Collaborative Information Visualization

The InfoVis community has also been interested in collaborative data exploration and analysis. Brennan et al. [2006] proposed a distributed collaborative visual analytics framework where individual group members have distinct system perspectives, or viewpoints. Group members have access to a shared knowledge base, and within their private viewpoints members can flexibly generate their own visualizations of information from the knowledge base. These private viewpoints and the visualization that they contain can be explicitly shared between group members in a *common ground*, a joint perspective that algorithmically merges multiple individual viewpoints. Merging operations such as morphing and fusing are provided to create the common ground perspective. Morphing identifies similarities between views and fusing generates aggregate viewpoints. Within Brennan et al.'s framework, the trade-off between individual and group work is handled by offering each group member an independent perspective, suitable for his/her own individual work, while at the same time providing support for transitioning these perspectives to more group orientated activities. This trade-off is similarly addressed in another distributed analysis system by Keel [2006]. Here, computational

agents were used to identify when an individual had uncovered potential relationships between information items in his/her workspace; this insight was then automatically relayed to the larger group of collaborators.

The systems proposed by Brennan et al. [2006] and Keel [2006] share admirable goals of providing explicit logical and graphical support for sharing information and translating among different views. However, as a collaborative visualization environment, emphasis is placed on individual perspectives (and thereby individual work) and spontaneous interactions are not easily possible. Systems should support the coordination of views and interactions so that team members can follow a momentary insight, glance at another's views, and transition quickly and effortlessly between different views of the data. This goal has been examined in two previous systems for co-located data analysis [Isenberg and Carpendale, 2007; Isenberg and Fisher, 2009]. In a co-located collaborative tree comparison system [Isenberg and Carpendale, 2007], group members could create multiple view instances and interact with these as separate entities, thereby enabling concurrent individual work. Yet, coordination among views was limited to a tree comparison operation and did not allow for group members to easily coordinate data annotations or other data modifications, and therefore, support of concurrent group work was limited. In Cambiera [Isenberg and Fisher, 2009], group members were provided with coordinated visualizations of their search results through text document collections. Through *collaborative brushing and linking* individual search results and text document representations were linked and joint interactions and search overlap were explicitly visualized.

2.6.4 Design Guidelines for Co-located Collaborative Information Visualization Systems

Isenberg and Carpendale [2007] characterized a series of design guidelines for *co-located collaborative information visualization systems*. These guidelines where assembled from a collection of advice drawn from the areas of information visualization design, co-located collaboration research, collaborative visualization studies, and observational studies on collaborative problem solving using information visualizations. This is precisely aligned with the research scope of my thesis work, and therefore, in this section I present a synopsis of these design guidelines. The original set of design guidelines, as outlined by Isenberg and Carpendale, are comprised of three main categories:

hardware and system setup, designing the information visualization, and designing the collaborative environment. These categories are further subdivided and my overview will follow this organization.

2.6.4.1 Hardware and System Setup

The first category of design guidelines looks at the configuration of the physical workspace from a technical perspective. This group of guidelines highlights the important issues that should be considered when determining the *size*, *configuration*, *input*, and *resolution* of the collaborative visualization environment.

Size: Effective collaboration using information visualizations requires a device that offers sufficient screen space to display the visualized information, as well as, allowing these visualizations to be viewed and shared by several group members. The size of the display workspace for supporting effective collaborative work has also been identified as an important issue by Scott et al. [2003a]. In this earlier work, Scott et al. asserted that the workplace must be large enough to allow group members to work comfortably with one another, without forcing people to encroach on each other's intimate space for prolonged periods. Isenberg and Carpendale's guidelines add an information visualization perspective to this requirement, asserting that there must be sufficient screen space available for each group member to view and share the visualized information.

Configuration: Numerous possibilities exist for hardware configurations of information displays, from large, interactive single-displays, such as a interactive tabletop displays or display walls, to individual interconnected displays; for example, the ConnecTable system [Tandler et al., 2001]. Different types of displays offer different affordances; therefore, the selected display has implications as to the type of visualization which can be most optimally presented, in addition to the type of collaborative work that is most optimal for that particular setup. Determining which display configuration to use should depend on that collaborative task and the type of group which will undertake this task.

Input: Scott et al. [2003a] asserted that concurrent interaction is a natural behaviour commonly observed in groups of people working together around traditional tabletops. To support this in a

digital environment, it would be ideal if collaborative visualization systems where to provide each group member with at least one means of input. Furthermore, it would be advantageous if these inputs were identifiable, enabling contextualization of system interaction with information as to who is performing the action.

Resolution: The resolution guideline is concerned with the the display resolution of the visualization's output device, as well as the resolution of the input device(s), as both of these are important issues for information visualization systems. Display resolution must be sufficiently high, such that images and text can be clearly discerned, and "visualizations might have to be re-designed if readability of text, color, and size is affected by display resolution" [Isenberg and Carpendale, 2007]. The pixel density of the display is another related issue that can be influential depending on the proximity of a person's interaction with the device. Close proximity to the displays requires a much higher pixel density, when compared to displays that are interacted with from afar. With interactive displays, input resolution can also be an issue, since direct touch input technology can often be too coarse to select individual pixels on a display with high pixel density. This is especially true when interacting with fingers. The issue is exasperated when working with information visualization systems, which typically have relatively small visual items that need to be directly interacted with.

Interactive Response: Information visualizations systems must be efficiently implemented such that systems perform at interactive rates, especially when handling numerous simultaneous inputs at any given time. It is important that visualization designers and developers use graphics techniques that ensure fast rendering of visualizations, supported by efficient software architecture design.

2.6.4.2 Designing the Information Visualization

Designing collaborative information visualizations draws on much of the previous guidelines suggested for non-collaborative visualization systems (e. g., [Ware, 2004; Tufte, 2001]). In this category, particular attention is paid to the issues which must be considered when designing for collaborative environments.

Supporting Mental Models: Flexible, user-controlled spatial arrangement and orientation of objects in a co-located collaborative workspace is crucial for facilitating group interactions and task coordination [Scott et al., 2003a, 2005]. Isenberg and Carpendale suggest that "letting users impose their own organization on items in the workspace may help collaborators *create and maintain mental models* of a data set that contains several different representations". Collaborative visualization environments should therefore allow group members to freely move objects in their workspace to whatever configuration they decide is most appropriate for a given task.

Representation Changes: Different information representations can provide different affordances for cognitive behaviours [Zhang and Norman, 1994]. For example, most people can calculate the product of two numbers easier when the numbers are represented in the decimal numeral system (e. g., 21×5), rather than the binary numeral system (e. g., 10101×101), although both representations provide the same information. Within cognitive science, this is referred to as the *representational effect*, "the phenomenon that different isomorphic representations of a common formal structure can cause dramatically different cognitive behaviors" [Zhang and Norman, 1994]. Different representations of common information therefore "provides different task efficiencies, task complexities, and changes decision-making strategies" [Isenberg and Carpendale, 2007]. Different representations can also present challenges in a collaborative environment. While making multiple representations available to group members may facilitate individual tasks, it can also hinder the larger group's communication concerning these objects that may potentially have varying representations [Gutwin and Greenberg, 1998]. As in the above numerical representation example, it might be difficult for collaborators to identify that 21×5 is the same as 10101×101 on first glance, however, if a single, commonly understood numerical representation was utilized, this identification would be significantly easier. Isenberg and Carpendale [2007] suggest that algorithmic support for the identification of common data items in a collaborative workspace might be a viable method in bridging the gap between multiple different representations of common information entities. The latter part of their paper presents a system which investigates the practicality of this approach for collaborative tree

comparison.

Task History: Providing history of information visualization interaction has been shown to facilitate iterative analysis in non-collaborative systems [Heer et al., 2008; Shneiderman, 1996]. In a collaborative environment, making interaction history available might be even more important, as group members may lose track of the interactions of their colleagues when transitioning between varying degrees of collaborative coupling across different tasks. Therefore, providing a graphical overview of task history "can help in later discussing the data and exploration results with collaborators or informing them about interesting data aspects that have been found during the analysis process" [Isenberg and Carpendale, 2007].

Perception: The effect of different display configurations on the interpretation of information visualizations is a relatively unexplored area of study. The perceptual scalability of visualizations presented on small and large display sizes has been studied by Yost and North [2006]. Based on the results of a controlled experiment, Yost and North suggested that for ensuring scalable visualizations, designers should consider:

- the effect of viewing distance and angle on visual encodings,
- using scalable graphical encodings within visualizations
- on large displays, provide both local and global visualization legends, and
- strategically placing labels in multiple locations throughout the visualization workspace.

Viewing angle was further studied by Wigdor et al. [2007], who looked at the affect of distortion on the perception of basic graphical elements of information visualizations, seen from different angles on a tabletop display. In general, it was found that some graphical elements are more affected by distortion than others, and this must be considered when selecting a distortion robust visual encoding. In collaborative visualization environments, such as interactive tabletop displays, group members are likely to be gathered around the device; thus, viewing the workspace from different directions. Designers should be cognizant of this issue, and therefore consider the affects of perception on the legibility of information visualizations.

2.6.4.3 Designing the Collaborative Environment

Guideline for effective design of collaboration visualization environments, draws on previous CSCW research into the basic operations that should be supported by shared-workspace groupware systems, in facilitating group members performing their tasks as a team [Pinelle et al., 2003]. These operations can be grouped as either *coordination* or *communication* aspects of the collaborative process.

Coordination: Group work requires the coordination of individual group members actions. Support for the coordination of activities can be approached from the following guidelines.

Workspace Organization: On traditional tabletops, group workspaces are typical composed of distinct personal, group, and storage territories [Scott et al., 2004]. Collaborative visualization environments should provide a flexible approach to workspace organization, such that these separate territories can organically emerge. Furthermore, a *group interaction and viewing space*, where shared representations and tools can be accessed by the group, is necessary. This area should be complemented with a *personal space* where group members can perform individual analysis activities, separate from the group.

Fluid Interaction: This guideline echoes the assertion from Scott et al. [2003a] that transitions between activities should be performed in a fluid manner, imposing little or no overhead. Providing support for fluid interaction thereby improves the coordination of activities.

Information Access: In a collaborative environment, the individual actions of group members must be coordinated at an information access level, at both global and local scopes. Modifications to shared objects within the workspace have the potential to affect the separate, individual activities of group members. Therefore, information access must be considered when looking to support coordination between group members in a collaborative setting.

Collaboration Styles: As noted in the previous discussion of mixed-focus collaboration (see Section 2.6.1), group members frequently transition between different degrees of collaborative cohesion. Collaborative environments must therefore support these different collaboration styles, which span from loosely coupled individual work, to closely coupled

group work. Individual work practices can be supported by providing multiple copies of objects within the workspace, such that collaborators can engage in individual, parallel activities with their own copies of an object. Group work practices can benefit from single shared instances of objects, which collaborators can access concurrently as a group. Related to the "supporting mental models" guideline, the ability to flexibly reorganize workspace objects also facilitates changing collaboration styles, as group members can change their proximity to one another, bringing their work items with them as they transition between different collaboration styles.

Communication: Successful collaboration depends on communication. In the mechanics of collaboration, Gutwin and Greenberg [1998] identified that both explicit and consequential communication are critical and these mechanics should not be interfered with. Furthermore, "people need to be able to trigger conversations, communicate about intentions to change collaboration styles, indicate a need to share a visualization, and to be generally aware of their team members' actions" [Isenberg and Carpendale, 2007].

In summary, the design guidelines by Isenberg and Carpendale [2007] assembles together much of the previous research for the design of information visualizations and co-located collaborative work environments, with a specific focus on the interplay of these two areas. These guidelines identify the specific issues that must be addressed when working in this design space, and therefore, this work is germane to my thesis research which operates within this very same design space.

2.7 Coordinated Multiple Views

Moving away from the collaboration discussion, I now introduce *coordinated multiple views (CMV)*, an important family of techniques for combining the interactions on multiple individual visualizations in a unified fashion. The general technique is characterized by simultaneously visualizing a data set in multiple different ways, where the individual visualizations, or *views*, are linked to one another via an interactive dependency [Weaver, 2006b, p. 17]. For example, Figure 2.10 shown previously, presents four different hierarchical data visualizations of the same data set (a multiple view visualization). One possible means of coordinating interactions across these individual views is with *brushing*

and linking: if a tree vertex in one view is selected, the same vertex is selected across all other views (a CMV visualization). More generally, *brushing* is an interaction technique in which graphical elements in an interface can be highlighted, selected, or deleted through direct pointing with an input device, such as a mouse [Ward, 1994]. In a multiple view environment, brushing is often combined with *linking*, where the same selected items in one view are selected across all views of that particular data set [Ward, 1994]. This CMV technique was first introduced over two decades ago by Becker and Cleveland [1987] and has since become a core feature in modern graphical computer interfaces. Brushing and linking is just one example of a possible coordination mechanism between multiple views of a common data set. Other mechanism have been discussed [North and Shneiderman, 1997; Shneiderman et al., 2009, §11.6.1].

Design guidelines for the effective use of multiple views have been outlined by Baldonado et al. [2000]; these guidelines are listed in Table 2.2. Baldonado et al. [2000] acknowledge that multiple view techniques are not always appropriate for a particular application; therefore, the first category of guidelines present design rules suggesting when the use of multiple views are warranted. These rules identify situations when multiple views are a particularly effective visualization technique. The second category of design rules looks at how multiple views should be used, recommending important design considerations which can facilitate in system usability.

The use of CMV is motivated by the assertion that an analyst can arrive at a better understanding of a data set when he/she is able to visualize the data from multiple complementary perspectives [Roberts, 1998; Pattison and Phillips, 2001]. Multiple representations of a common data set also "provides different task efficiencies, task complexities, and changes decision-making strategies" [Isenberg and Carpendale, 2007]. Overloading a single visual representations with a large number of visual variables can quickly lead to an overwhelming cognitive load for anyone attempting to engage with the system. This cognitive load can be reduced by using multiple, simple representations, even when including the effort required to manage these multiple views [Pattison and Phillips, 2001]. To ease in the navigation and understanding of an information space that is spread out across multiple views, interactions are coordinated across the views. Coordinate visualizations introduce their own set of design trade-offs, as a visualization environment with a simple set of coordination

	Rule	Summary	Major Impacts on Utility: Positive	Major Impacts on Utility: Negative
When To Use	Diversity	Use multiple views when there is a diversity of attributes, models, user profiles, levels of abstraction, or genres.	· memory	· learning · computational overhead · display space overhead
	Complementarity	Use multiple views when different views bring out correlations and/or disparities.	· memory · comparison · context switching	· learning · computational overhead · display space overhead
	Decomposition	Partition complex data into multiple views to create manageable chunks and to provide insight into the interaction among different dimensions.	· memory · comparison	· learning · computational overhead · display space overhead
	Parsimony	Use multiple views minimally.	· learning · computational overhead · display space overhead	· memory · comparison · context switching
How To Use	Space/Time Resource Optimization	Balance the spatial and temporal costs of presenting multiple views with the spatial and temporal benefits of using the views.	· comparison · computational overhead · display space overhead	
	Self-Evidence	Use perceptual cues to make relationships among multiple views more apparent to the user.	· learning · comparison	· computational overhead
	Consistency	Make the interfaces for multiple views consistent, and make the states of multiple views consistent.	· learning · comparison	· computational overhead
	Attention Management	Use perceptual techniques to focus the user's attention on the right view at the right time.	· memory · context switching	· computational overhead

Table 2.2: Design guidelines for the effective use of multiple views [from Baldonado et al., 2000, p. 118].

mechanisms might be easy to understand, yet lacking in expressivity and customizability. On the other hand, the environment becomes more powerful, albeit more complex, when the number of views increases and a large customizable set of relationships for coordinating interactions on these views is made available [Weaver, 2006a].

Work by North and Shneiderman [2000a] and Weaver [2004] has investigated the design space of rich, customizable CMV systems, exploring their utility [North and Shneiderman, 2000b] and techniques for balancing the associated design tradeoffs. North and Shneiderman's "Snap-Together Visualization" system is conceptualized using a relational database model, where coordinate visualizations are constructed by relational joins on the database schema. This elegant solution empowers analysts with the ability to interactively construct his/her own CMV environment, using familiar concepts from relational databases. Weaver's "Improvise" system is also modelled after a relational database using a "flexible, expression-based visual abstraction language" [Weaver, 2004] for creating customizable coordinate visualizations.

2.8 Meta-visualization

Full featured CMV systems, like those demonstrated by North and Shneiderman [2000a] and Weaver [2004], offer a rich set of customizable coordinations between multiple views. These systems are indeed powerful, however they can become overly complex, making the visualization difficult to comprehend [Weaver, 2006b, p. 171]. To understand a CMV visualization, the analyst working with the system must be able to conceptualize the coordination system which connects the individual views [Weaver, 2006b, p. 170]. As the number of views and the coordinations between views increases, this task becomes increasingly more arduous.

To manage this complexity, Weaver [2005] introduced a *meta-visualization* to help people conceptualize the coordination structure in the Improvise system. A *meta-visualization* is a "visualization of another visualization's structure and operation" [Weaver, 2006b, p. 172] and can be used in CMV systems to make the coordination relationships between views visually explicit. Improvise realizes meta-visualizations ranging from embedding/overlaying the meta-visualization directly on the individual views, to providing a separate view which displays the meta-visual data. Figure 2.11 illus-

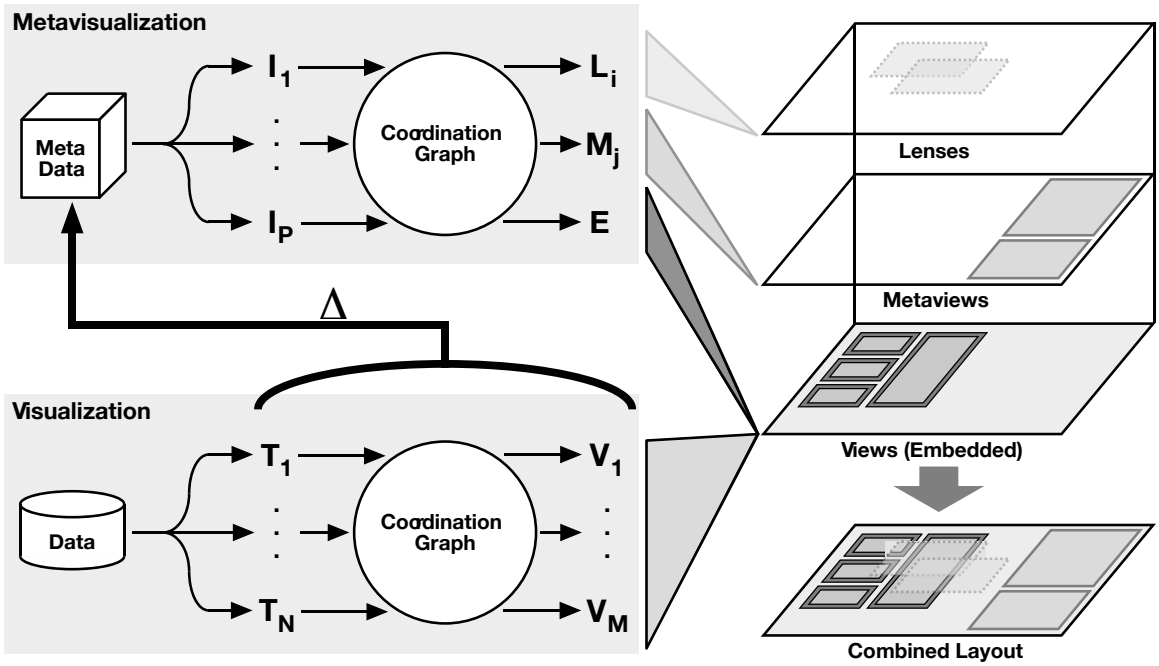


Figure 2.11: An integrated meta-visualization model [from Weaver, 2005, p. 166]. In a CMV system, the visualization is comprised of data subsets (T) that are linked together using a coordination graph, producing a series of views (V). This information makes up the metadata. The meta-visualization takes the dynamic representations (I) from the meta-data, links them together using a coordination graph, producing the meta-visualizations: coordinated lenses (L), metaviews (M), and meta-visual embedding (E). *Weaver, Visualizing Coordination In Situ, Proceedings of the IEEE Symposium on Information Visualization, © 2005 IEEE. Included here by permission.*

trates Weaver's conceptual model for an integrated meta-visualization. Figure 2.12 and Figure 2.13 present examples of how this model is applied within Improvise. These example meta-visualizations convey a lot of information that would otherwise be tacit, left to the analyst to manage. In a full featured CMV system like Improvise, visualizing the coordination relationships alleviates a significant amount of cognitive overhead that is now externalized and accurately matches what is happening in the visualization system.

Another meta-visualization example is the VisLink CMV system by Collins and Carpendale [2007]. In this work, links are drawn between common data items visualized using different representations, as shown in Figure 2.14. VisLink is a low-level technique when compared to the meta-visualization in Improvise, as the focus is on the occurrence of individual data items across multiple

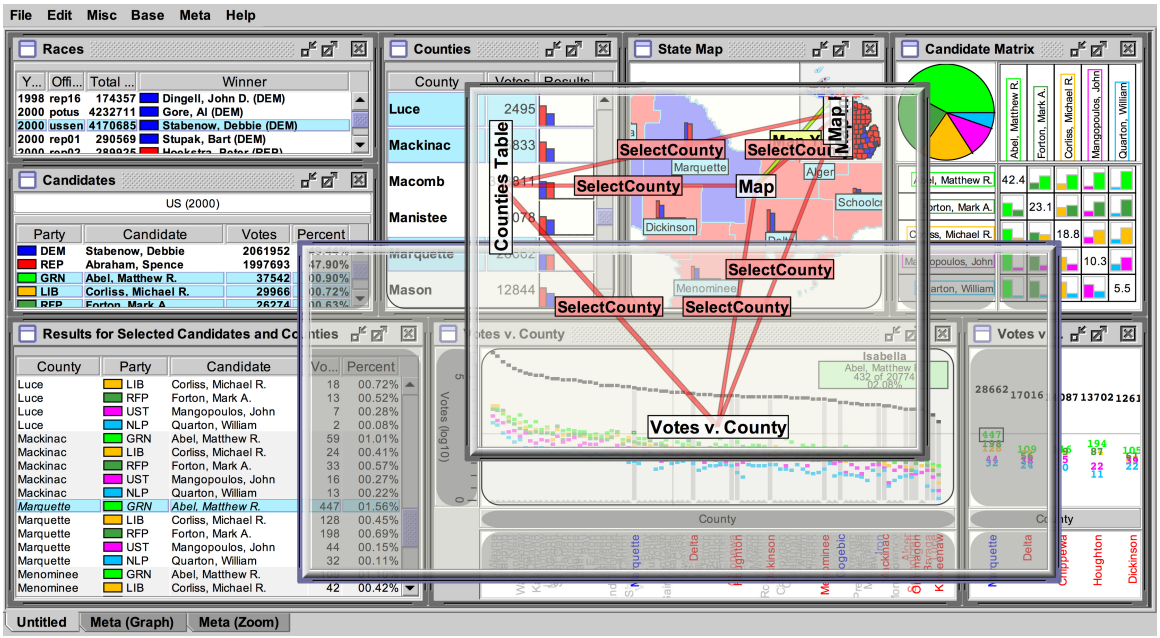


Figure 2.12: A view of Improvise's interface, visualizing election results from the State of Michigan from 1998 to 2004 [from Weaver, 2005, p. 167]. Here we see the CMV visualization environment with two meta-visual lenses overlaid across workspace. The upper most lens shows the coordination structure between four views: Counties Table, Map, Map Inset, and Votes versus County. A second, lower lens shows which views are selected, illustrated using round rectangles of different shades of grey. The lenses are bevelled and tinted to help distinguish them from the visualization below. *Weaver, Visualizing Coordination In Situ, Proceedings of the IEEE Symposium on Information Visualization, © 2005 IEEE. Included here by permission.*

visualizations, rather than presenting metadata about the coordination structure between visualizations.

2.9 Summary

In this chapter, I have presented the body of previous research that forms the foundation of my own thesis research. My overview began with defining the general field of visualization and characterizing two prominent disciplines: SciVis and InfoVis. I then introduced how these disciplines have conceptualized the visualization process with the creation of the data flow and the date state process models. Particularly close attention was paid to the operator interaction framework created by Chi and Riedl [1998], outlining the theoretical considerations that this framework seeks to address. Moving from theoretical to pragmatic territory, I then introduced a mathematical definition of hier-

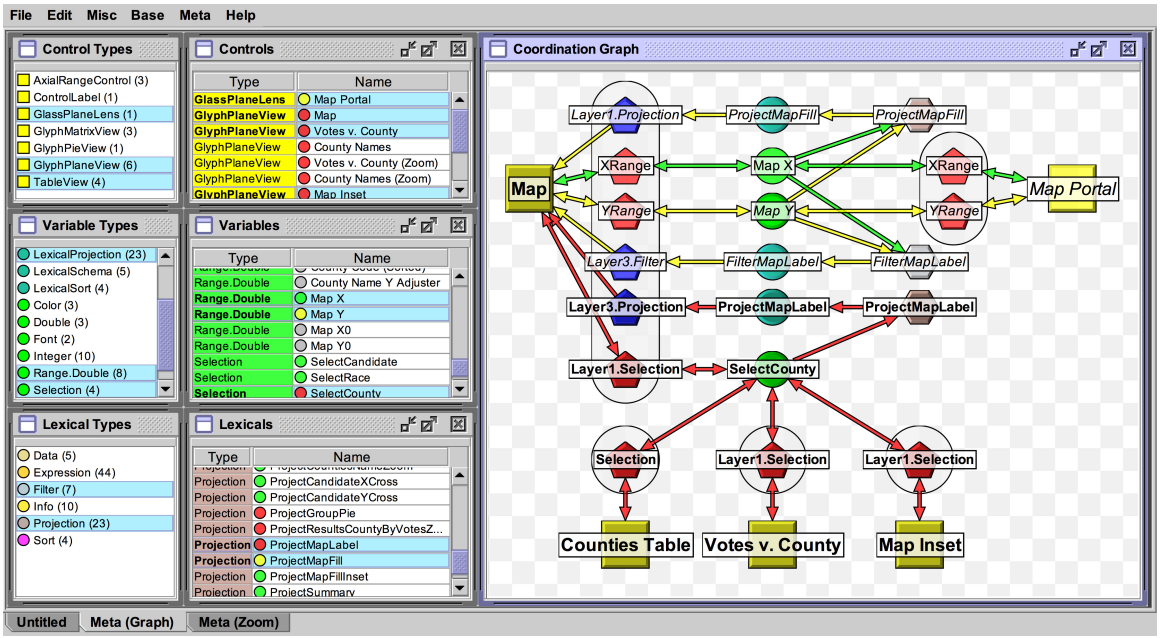


Figure 2.13: In a separate window tab, a coordination query graph illustrates how the individual views shown in Figure 2.12 are connected to one another within Improvise [from Weaver, 2005, p. 169]. In this metaview, coordinations between views can be added and removed interactively. A series of lists present an overview of the visualization's constituent parts and their interactive state (green for inactive, yellow for focused, and red for editing). *Weaver, Visualizing Coordination In Situ, Proceedings of the IEEE Symposium on Information Visualization, © 2005 IEEE. Included here by permission.*

archical data and an overview of some of the common visualization techniques for representing this type of data. Next, I introduced previous research on collaboration in synchronous co-located environments. Specifically, I focused the discussion around small groups of people involved in mix-focus co-located collaborative data analysis that makes use of information visualizations on a interactive tabletop display. This examination of collaboration research was followed by a discussion of CMV techniques. Lastly, I introduced the use of meta-visualizations in CMV environments. Building off of this previous research, in Chapter 3, I present the collaboration concept behind the Lark system.

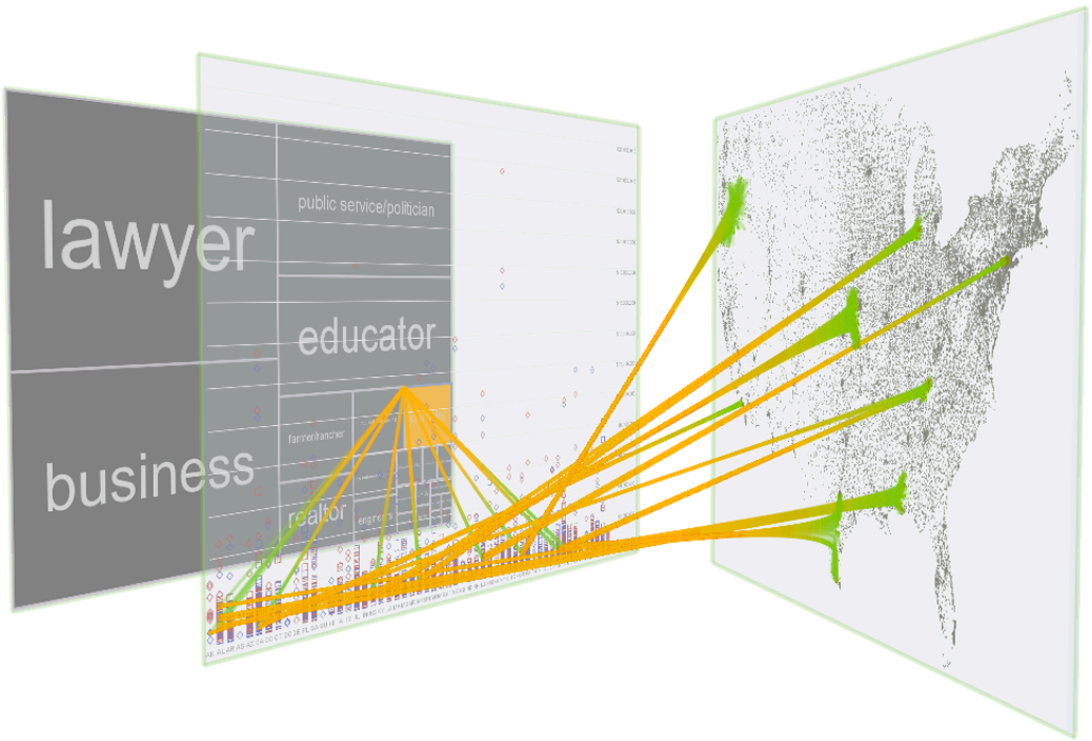


Figure 2.14: The VisLink CMV system [from Collins and Carpendale, 2007, p. 1198]. VisLink uses brushing and linking, with a meta-visualization drawing links between selected items across the three visualizations—treemap (left), scatterplot (middle), and geographic map (right). *Collins and Carpendale, VisLink: Revealing Relationships Amongst Visualizations, IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization), © 2007 IEEE. Included here by permission.*

Chapter 3

Lark: Collaboration Concept

Lark is a system that facilitates the coordination of collaborative interactions with information visualizations on shared digital workspaces. Here I introduce through three distinct perspectives: conceptual design, interface design, and software architecture design. Each of these perspectives are discussed in their own individual chapter, beginning here in Chapter 3 with the conceptual design. In this discussion I explain the high level concepts upon which Lark is built, drawing on previously introduced topics from the review of related work in Chapter 2.

This chapter is organized as follows. Section 3.1 presents the background surrounding the design process of Lark. Before moving into the design process in more detail, Section 3.2 briefly introduces the Lark system, providing some grounding to the conceptual discussion that follows. Section 3.3 outlines the design challenges and the design decisions identified during the research and development of Lark, summarized in Figure 3.1. The discussion is structured by connecting Lark's design process with the design guidelines for co-located collaboration information visualization systems, as outlined by Isenberg and Carpendale [2007], and the design guidelines for multiple view system by Baldonado et al. [2000], as well as the principles identified in the literature review from Chapter 2. The chapter is concluded with Section 3.5 which summarizes the conceptualization of Lark.

3.1 Background

The context for the research presented in this thesis is a collaboration between a group of Information Visualization (InfoVis) researchers—comprised of myself, Petra Isenberg (a PhD student at the time), and our supervisor Dr. Sheelagh Carpendale—and a group of research biologists lead by Dr. Michael G. Surette from the Department of Microbiology and Infectious Diseases, in the Faculty of Medicine at the University of Calgary. This collaboration grounded this research in real-world biological data, tasks, and context. My InfoVis colleagues and I were interested in investigating the collaborative processes already being practised within this group of biologists, and studying how their collaborative practices might be better facilitated by digital tools that have been designed with their collaborative processes in mind, particularly for data analysis tasks. To this end, we conducted a series of informal interviews and walkthroughs of the current analysis processes employed by a select group of individual biologists when undergoing empirical data analysis. These accounts of individual work flows and processes where then juxtaposed by informal observations made during regular attendance of the biologists' weekly research meetings, where research progress, experimental hypotheses and results, and data analysis strategies were discussed by the group. These weekly meetings demonstrated numerous instances of collaborative work occurring in the group. The observations made therein where used to inform the design of Lark, a visualization system for collaborative data analysis.

Our informal observations of the biologists' collaborative processes suggested that one of the most necessary pieces of technology needed to support collaboration is *not* novel data analysis tools, but better support for their collaborative practices. Thus what would be more appropriate in addressing their challenges is collecting together existing, disparate tools into a unified data analysis environment designed to support collaboration. As a result, instead of new tools to facilitate individual work, my research focuses on designing a system which supports collaboration during group analysis tasks; exploring software to support changing collaboration styles. Therefore, while this thesis is set in a biological context, the system's primary focus is exploring ways to support collaboration.

3.2 Overview of Lark

Lark is a co-located collaborative information visualization environment in which an integrated meta-visualization [Weaver, 2005] shows the links and relationships between multiple coordinated views. Traditional coordinated multiple views (CMV) systems propagate the mapping of actions on objects in one visualization to actions in another [North and Shneiderman, 2000a]. Lark extends this general feature to better support coordination of synchronous interactions from multiple people collaborating in a shared workspace; which has been identified as an important feature in supporting collaborative work within a digital context [Scott et al., 2003a]. To support the coordination of interactions, Lark's meta-visualization is modelled after the conceptual visualization pipeline, which provides several distinct stages in which group members can coordinate their actions. In doing so, Lark extends existing CMV [North and Shneiderman, 2000a; Weaver, 2004] and meta-visualization [Weaver, 2005; Collins and Carpendale, 2007] techniques with its novel approach to supporting mixed-focus collaborative work.

Enabling concurrent interaction with objects in a shared digital workspace requires careful consideration of how to provide tools such that group members can coordinate their own actions, without interference from the system. Viewing this challenge from the perspective of a visualization process model, we can see that this coordination can take place at several different stages of the visualization pipeline—data, representations, presentation, or view level [Heer and Agrawala, 2008]. Making the stage at which coordination is taking place visually explicit may help provide awareness, which collaborators can use to flexibly coordinate their activities. Lark's information visualization environment supports this type of coordination by integrating a representation of the visualization pipeline into the shared workspace, thus indicating coordination points for data, representation, presentation, and view levels. This meta-visualization of the underlying visualization process makes the connections and relationships between the individual views visually explicit, providing workspace organization and awareness. The underlying goal is to support mixed-focus collaborative work. Lark strives to realize this goal by supporting varying levels of collaborative cohesion—from loosely coupled, individual work to closely coupled, group work—as well as, the integration of and transitions between these different collaboration styles.

3.3 Design Process

Within the collaborative work scenario that I am interested in (introduced in Section 1.2), I centre my research effort on supporting mixed-focus collaboration. Providing a means to facilitate changing collaboration styles is fundamental to supporting mixed-focus collaboration. To this end, I dicsuss the *collaboration and coordination concept* underpinning Lark which focuses on supporting changing collaboration styles. This conceptual design draws on system design guidelines for co-located collaborative information visualizations [Isenberg and Carpendale, 2007] and multiple views [Baldonado et al., 2000], introduced previously in Section 2.6.4 and Section 2.7 respectively. The design guidelines by Isenberg and Carpendale [2007] were assembled from a collection of advice drawn from the areas of information visualization design, co-located collaboration research, collaborative visualization studies, and observational studies on collaborative problem solving using information visualizations. This is precisely within the research scope of my thesis work, and therefore, these guidelines are particularly well suited for explaining the design process of Lark. Lark's information visualization environment makes use of multiple view techniques in visualizing hierarchical data. The design of this visualization environment makes use of previously published design guidelines for multiple view systems by Baldonado et al. [2000] to avoid known shortcomings when utilizing this family of visualization techniques.

The design process behind Lark's collaboration and coordination concept is illustrated in Figure 3.1. This design process overview is shown as an annotated sequence of steps, each categorized as either a *design challenge* or *design decision*, which incorporates the design guidelines from Isenberg and Carpendale [2007] and Baldonado et al. [2000]. Each step in the process is influenced by the previously identified design challenges and design decisions. The discussion of the design process follows this sequential iteration of design steps, beginning at the top of Figure 3.1 with *support changing collaboration styles*, and each step is discussed in its own subsection. To avoid having to constantly refer back to Figure 3.1, a portion of this figure has been recreated in each of the subsections to better contextualize that particular design step discussion.

In this context, a *design challenge* is defined as an aspect of the environment, task, *etc.* that has been identified as difficult to overcome and must be addressed directly in order to achieve the objec-

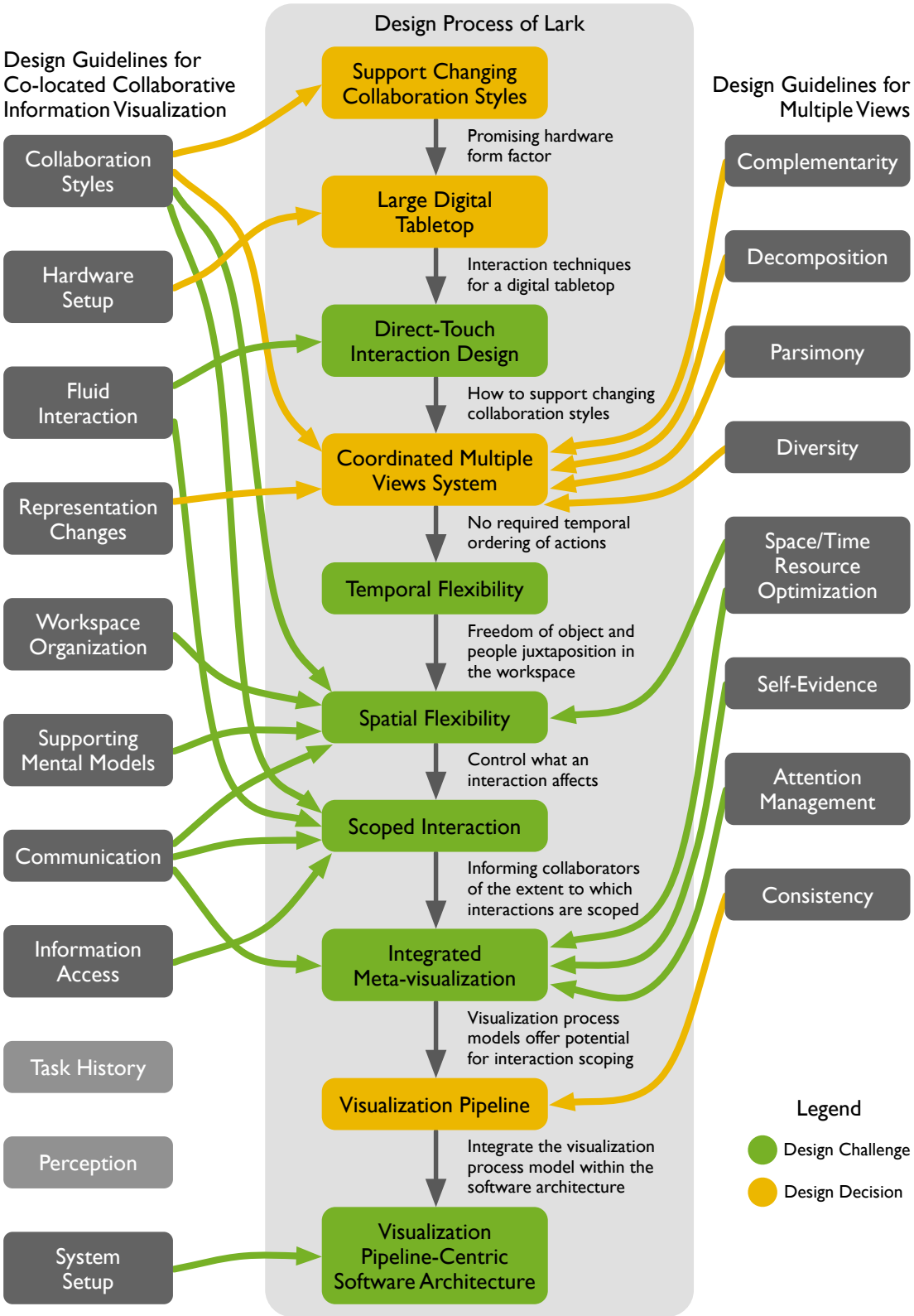


Figure 3.1: Lark's design process.

tives of a project. Explicitly outlining these challenges in the early design stages can help in identifying solutions as they arise later on in the design process. Furthermore, design challenges from one specific project can often be generalized and applied to other related projects within the same research domain addressing similar issues. A *design decision* is a declaration of a specific design choice made and the resulting actions taken during the design process. This choice may, for instance, be based on the identified design challenges, suggestions from existing literature on the topic, and/or insights gained from an iterative design process. The explicit declaration of design decisions act as bifurcation points, enabling future research endeavours to make conscious alternative choices at these points, leading to potentially different end results. The remainder of this section traces through the design process of Lark, explaining the design decisions that were made and discussing the design challenges that where identified.

3.3.1 Support Changing Collaboration Styles

The principal design decision in the conceptual development of Lark is to focus on *supporting changing collaboration styles*. It is from this point that Lark's design process begins (see Figure 3.1) and all other design choices are subordinate to this primary focus. Supporting changing collaboration styles deserves this attention as much evidence has been gathered about the importance of supporting team members in switching between tightly and loosely coupled collaborative work [Elwart-Keys et al., 1990; Mandviwalla and Olfman, 1994; Gutwin and Greenberg, 1998; Scott et al., 2003a; Isenberg et al., 2008]. While collaboration styles are frequently discussed as parallel and joint, it has been shown that these are end points of a continuum which exhibits many variations in the degree of cohesion between team members [Tang et al., 2006]. For example, a small group of people engaged in collaborative work with information visualizations could transition through the following phases:

A: Joint examination of one shared view, where everyone is interacting with the single view and collectively discussing their findings.

B: Joint examination of one shared view, where only one group member is interacting with the single view and the other members are observing.

C: Parallel exploration using different views, where each group member has his/her own individual

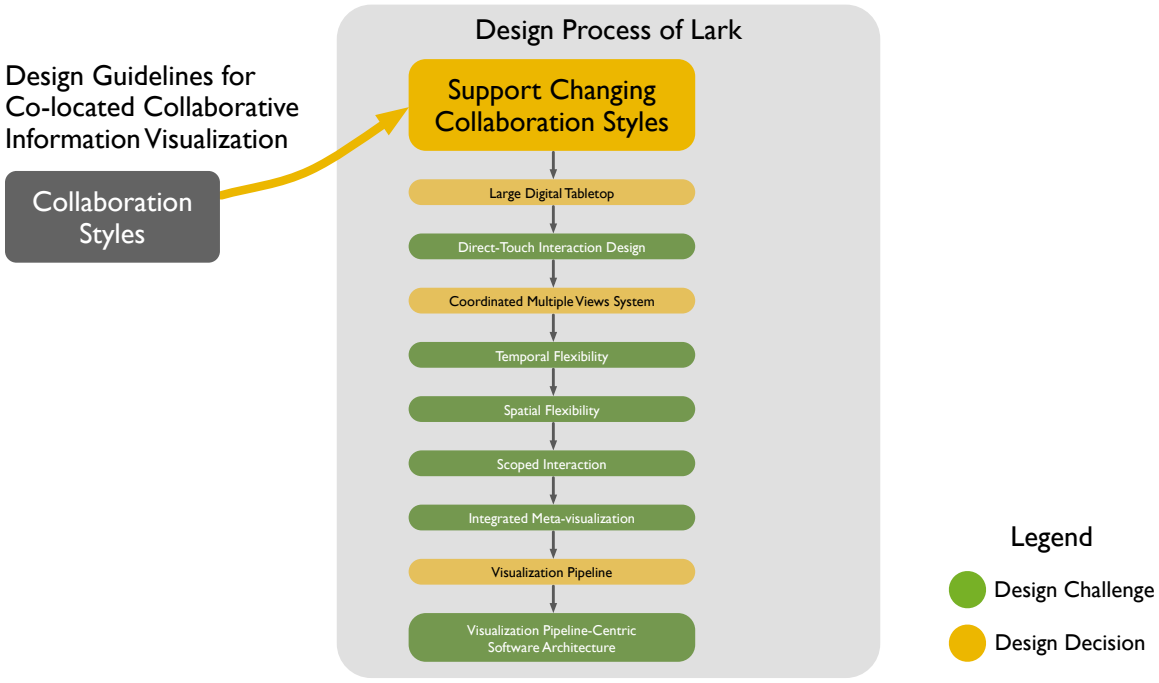


Figure 3.2: Lark's design decision to *support changing collaboration styles*.

view and are closely communicating about his/her findings with one another.

D: Parallel exploration using different views, where each group member has his/her own individual view and are working individually, with little explicit communication between the group.

Changing collaboration styles requires support for both group work practices (Scenario A) and individual work practices (Scenario D), as well as the gradual transition between the different styles of work (Scenario A through D). It is this type of support for collaboration that I strive to provide with Lark's visualization environment.

Design guidelines for *collaboration styles* by Isenberg and Carpendale [2007] (see Section 2.6.4.3) have suggested the following strategies for realizing the support of changing collaboration styles. Group work may be better facilitated by providing single shared instances of visualizations in the workspace, which group members can access concurrently (Scenario A). Individual work could be facilitated with the ability to create multiple copies of visualizations, enabling individual parallel activities (Scenario D). The transitions between group work and individual work should be supported by the ability to fluidly transform objects between these different states. For example, transitioning

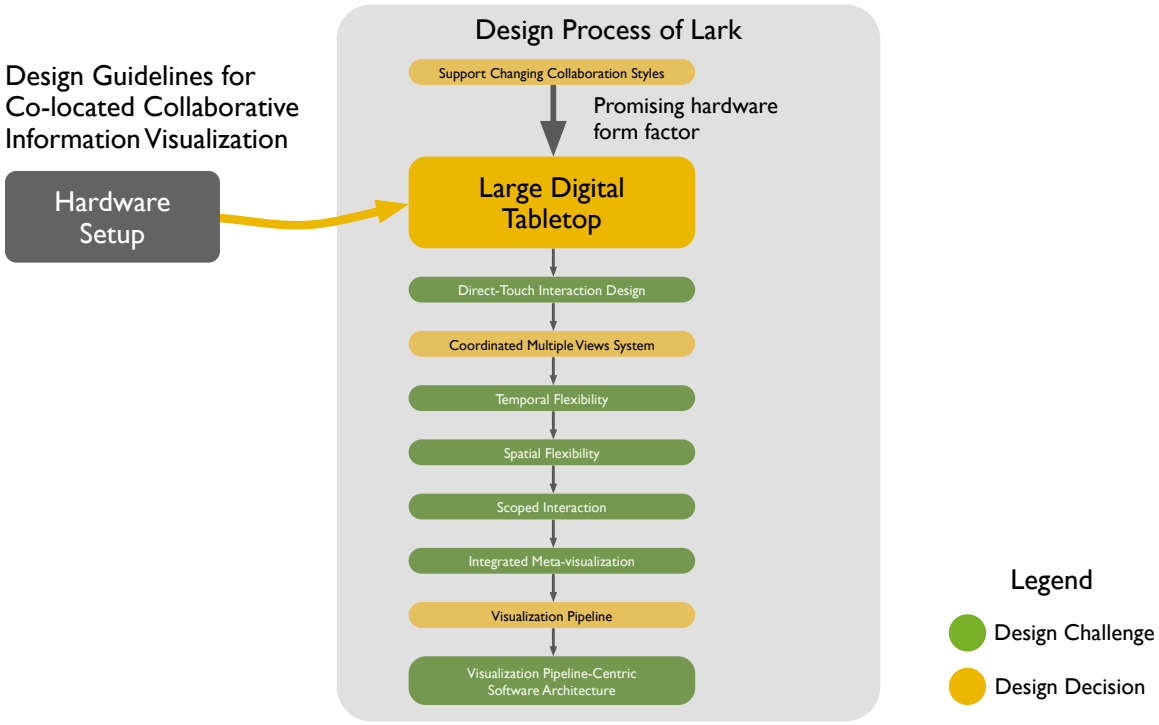


Figure 3.3: Lark's design decision to use a *large digital tabletop* as the hardware form factor.

from joint examination (Scenario A) to parallel individual work (Scenario D), group members could break off from their group task by creating individual instances of the shared visualization under study and continue the analysis individually with their own private visualization instances. This dynamic coordination of interactions in facilitating changing collaboration styles is precisely what I hope to achieve.

3.3.2 Large Digital Tabletop

A promising hardware form factor that is well suited for studying mixed-focus collaborative work are *large, interactive tabletop displays*. Previous work on mixed-focus collaboration [Rogers and Lindley, 2004; Tang et al., 2006], have shown that this form factor provides unique affordances conducive to this type of collaborative work, and therefore Lark's second design decision was to use this hardware configuration. Selecting digital tabletops as the computing form factor follows the *hardware setup* design guideline outlined by Isenberg and Carpendale [2007]. This guideline is comprised of five individual categories—configuration, input, size, resolution, and interactive response—introduced

Figure 3.4: The large, multi-touch tabletop display from SMART Technologies that was used in the development of Lark.

in Section 2.6.4.1 and each of which are touched on in the discussion below. It should be noted that I use the very same hardware configuration as that used by Isenberg and Carpendale [2007] in the development of their collaborative tree comparison visualization system.

Lark was designed for use on a large, interactive tabletop display. The development system consisted of a touch-sensitive 1.5 × 1 meter DViT Board from SMART Technologies [SMART Technologies ULC, 2010] which makes up the tabletop surface. The display size is sufficient for groups of two to four people to work comfortably together with enough area to share information visualizations [Isenberg and Carpendale, 2007]. Figure 3.4 presents a photograph of the tabletop display; the scene includes three people to illustrate the relative size of the device. The system is capable of capturing two non-identifiable concurrent and independent inputs, allowing two people to work synchronously with objects in the workspace. Support for only two concurrent inputs constrains the optimal number of group members who can simultaneously collaborate around this particular tabletop display, to groups of two people. This table was build circa 2005 and at the time was state-of-the-art, given its large display size and high resolution. Support for two concurrent inputs was the

best available technology, given the trade-offs with display size and resolution, as all other solutions were simply cost prohibitive for an academic research lab.

The tabletop display operates at a resolution of 2,800 × 2,100 pixels. This 5.9 megapixel display is provided by four rear-mounted projectors in a 2 × 2 configuration. In 2005, a projector operating at a resolution of 1,400 × 1,050 pixels was comparable to desktop displays at the time. The projectors are driven by two NVIDIA GeForce 7900 graphics cards [NVIDIA Corporation, 2010] connected via SLI running under Microsoft Windows XP on a single-core 3.0 GHz processor with 2 GB of RAM.

The minimum font size used in Lark's visualization was selected such that text can be easily read at the provided display resolution. The pixel density of the display is approximately 53 pixels per inch (PPI), which in 2005 was acceptable for close proximity interaction. The input resolution of the display, when using a finger or pen, is rather coarse given the high output resolution. Lark's interface has therefore been designed with sufficiently large selection areas on all interface widgets requiring direct touch interaction. However, selection of individual items contained within a visualization can be an issue. I have investigated this issue in other research pursuits [Voida et al., 2009], however a discussion of this work is beyond the scope of this thesis.

3.3.3 Direct-Touch Interaction Design

Using an interactive tabletop display as the hardware form factor has implications on Lark's interaction design. Rather than following traditional interaction techniques intended for mouse- and keyboard-based systems, making full use of the tabletop display demands shifting the focus to a *direct-touch interaction design paradigm* [Shen et al., 2006]. Lark's first design challenge is to utilize the unique affordances of the interactive tabletop medium and design the system's interaction techniques specifically for a direct-touch environment. Designing tabletop interaction techniques that can be performed with ease realizes the guideline of *fluid interaction* [Isenberg and Carpendale, 2007] (see Section 2.6.4.3). Lark's interaction paradigm makes exclusive use of direct-touch input, thereby avoiding the usability costs associated with shifting between input modes (e. g., moving from touch-based interaction to using a keyboard).

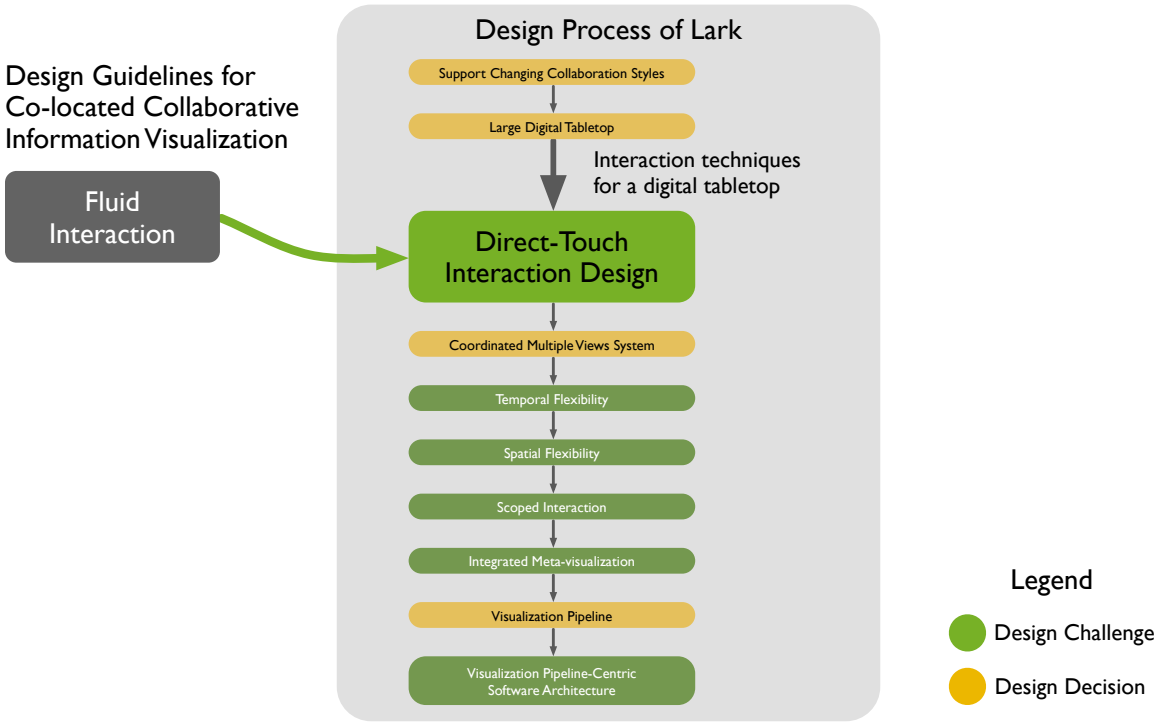


Figure 3.5: Lark's design challenge of *direct-touch interaction design*.

3.3.4 Coordinated Multiple Views System

With Lark's hardware setup and the associated design implications for interaction techniques determined, the next item in Lark's design process returns to the initial question of how to support changing collaboration styles. My research addresses this by providing a coordinated multiple views (CMV) environment within Lark's visualization system, supporting the visual analysis of hierarchical data. Previous work has suggested that a multiple view environment is well suited for mixed-focus collaborative work [Scott et al., 2003a]. The decision to use CMV is also supported by guidelines for both co-located collaborative information visualization [Isenberg and Carpendale, 2007] and the use of multiple views [Baldonado et al., 2000].

In Isenberg and Carpendale's guideline for *collaboration styles* (see Section 2.6.4.3), it is suggested that individual work practices can be supported by providing multiple copies of objects within the workspace, and group work practices can benefit from a single shared instance of an object. In the *representation changes* guideline (see Section 2.6.4.2), Isenberg and Carpendale identify the need for providing multiple representations of the same information, empowering group members to use

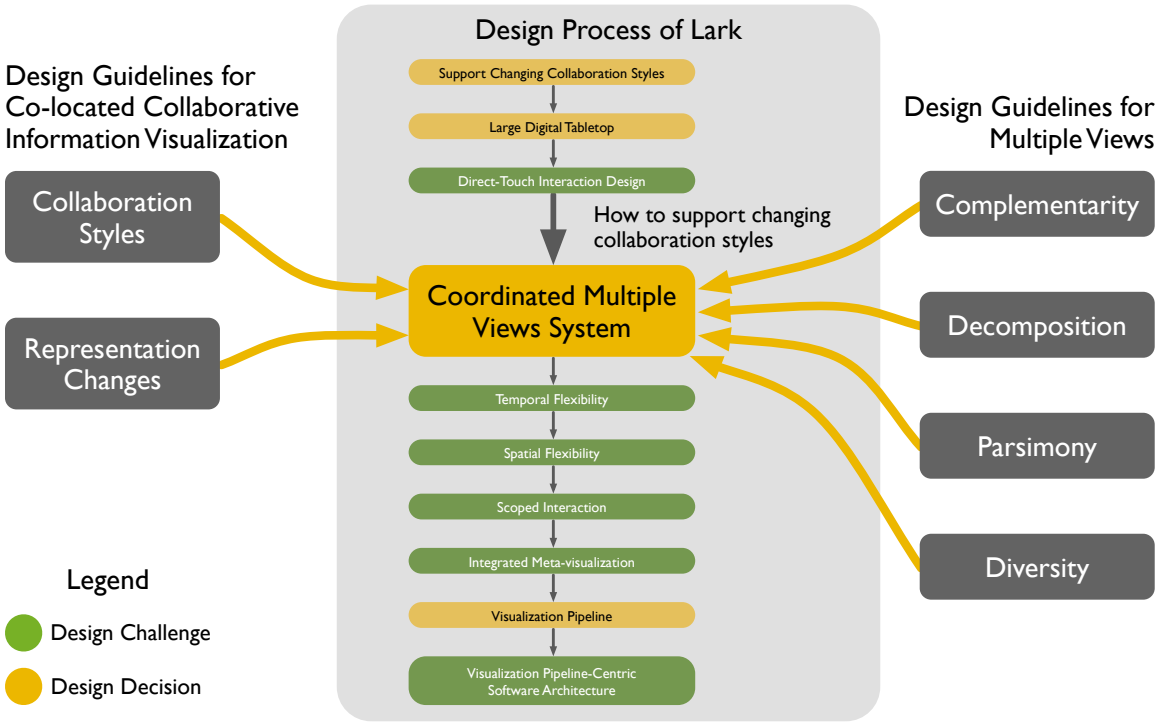


Figure 3.6: Lark's design decision to use a *coordinated multiple views system*.

whichever representation they feel is more appropriate for the task at hand. Both of these guidelines can be realized through the use of CMV visualization techniques, that is: providing multiple visualizations of a common data set, where these individual views are linked to one another via an interactive dependency [Weaver, 2006b]. The representation changes guideline can be followed by providing multiple options as to the visual representation of data. Lark's visualization environment supports this by visualizing hierarchical data through four types of representations: cladogram, radial cladogram, icicle plot, and sunburst (as illustrated in Figure 2.10). The collaboration styles guideline can be followed by providing a dynamic number of these views within the visualization workspace where the number of views is determined by the individual collaborators. Lark supports the dynamic creation of views, controlled by people rather then the system. An open question in using CMV is determining an appropriate coordination mechanism for connecting the multiple views to one another. My research addresses this question later on in Lark's design process, beginning with *scoped interaction* (see Section 3.3.7).

The design decision to use CMV is also supported by design guidelines from Baldonado et al.

[2000] identifying situations when multiple view techniques can be particularly effective. Our collaborative work scenario follows Baldonado et al.'s rules of complementarity, decomposition, parsimony, and diversity as introduced in Table 2.2. The *rule of complementarity* suggests that it is easier to compare visualizations when they can be juxtaposed alongside each other and compared visually, rather than having to rely on memory to make the comparison. Since comparison has been shown to be an important strategy for data analysis [Munzner et al., 2003; Isenberg and Carpendale, 2007], providing techniques that improve comparison efficiency is of value. Therefore, the use of multiple views within Lark is indeed warranted. The *rule of decomposition* characterizes handling complex data by breaking the whole down into manageable parts. This *divide and conquer* approach is often used in collaboration work situations [Isenberg et al., 2008] and Lark's work scenario conforms to this rule. The *rule of parsimony* suggests that multiple views should be used minimally. This is echoed in Lark's design as it is the individual collaborators who control the creation of views. Lastly, it is suggested that multiple views are appropriate when there is a *diversity of user profiles* (e. g., preferences, levels of expertise, roles) that are to be supported by the system. This again is appropriate for the work scenario I am investigating as collaborative data analysis leverages the analytic power of multiple individuals where these individuals often have varying types and levels of expertise.

3.3.5 Temporal Flexibility

The next step in Lark's design process is acknowledging that there should be no required temporal ordering of actions within the visualization environment. This design challenge embodies one of the large research challenges of this thesis, introduced in Section 1.5, which investigates *how to provide temporal flexibility for data analysis activities*. This challenge is not drawn from either group of design guidelines, but rather more recent work by Isenberg et al. [2008] who conducted an exploratory study of individuals and small groups working with paper-based visualizations. The study investigated how participants went about answering a series of assigned questions, organized into two separate scenarios. Analysis of the collected data identified eight common *information analysis processes* that were employed by participants: browse, parse, discuss collaboration style, establish task strategy, clarify, select, operate, and validate. It was observed that "the processes themselves were

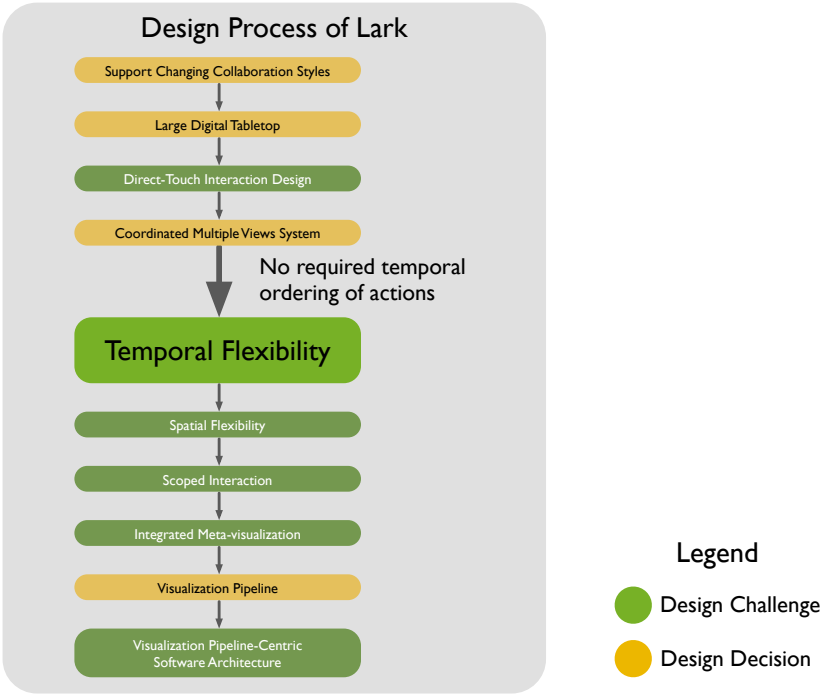


Figure 3.7: Lark's design challenge of *temporal flexibility*.

not temporally organized in a consistent way across groups" [Isenberg et al., 2008]. For example, what one group of collaborators did at the beginning of their analysis session might be performed halfway through a session by another group. Figure 3.8 presents an original image from Isenberg et al.'s paper illustrating the temporal sequencing of categorized information analysis processes for three pairs of participants during a complete scenario. Based on the study results, Isenberg et al. concluded that temporal flexibility of analysis processes is common practice among group members engaged in mixed-focus collaboration. Lark's second design challenge is to adhere to this finding, providing a visualization environment where analysis actions are temporally flexible, free of any pre-defined ordering of actions.

3.3.6 Spatial Flexibility

Spatial flexibility is the next challenge in the design of Lark. Like temporal flexibility, this design challenge embodies one of the larger research challenges of *how to provide spatial flexibility of visualization and collaborators around the shared workspace* (see Section 1.5). The formulation of this challenge ties back in with supporting changing collaboration styles, as changes in collaborative co-

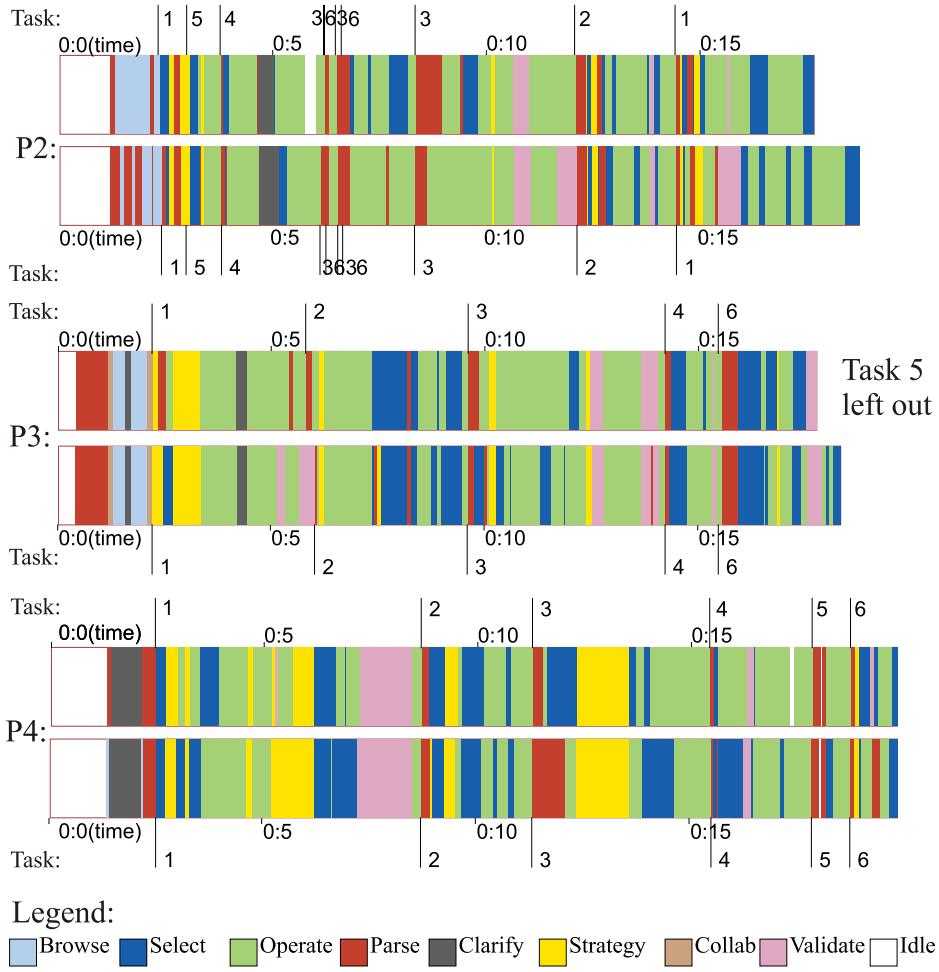


Figure 3.8: Results from a exploratory study of individuals and small groups working with paper based visualizations [from Isenberg et al., 2008, p. 1223]. This figure illustrates the temporal sequencing of the eight common *information analysis processes* for three pairs of participants during an analysis scenario. Notice the inconsistent ordering of processes across the different groups. *© 2008 ACM, Inc. Included here by permission.*

hesion in large workspaces is commonly accompanied by changing team member locations and the associated reorganization of workspace items [Scott et al., 2003a]. During individual work, team members may want to keep views they are working on in close proximity or within their personal territories [Scott et al., 2004]. Transitioning to more closely coupled work may require the relocation and resizing of views so that they can more easily be seen and accessed by multiple collaborators. Designing a workspace that supports this type of unconstrained spatial organization and mobility is the objective of this challenge.

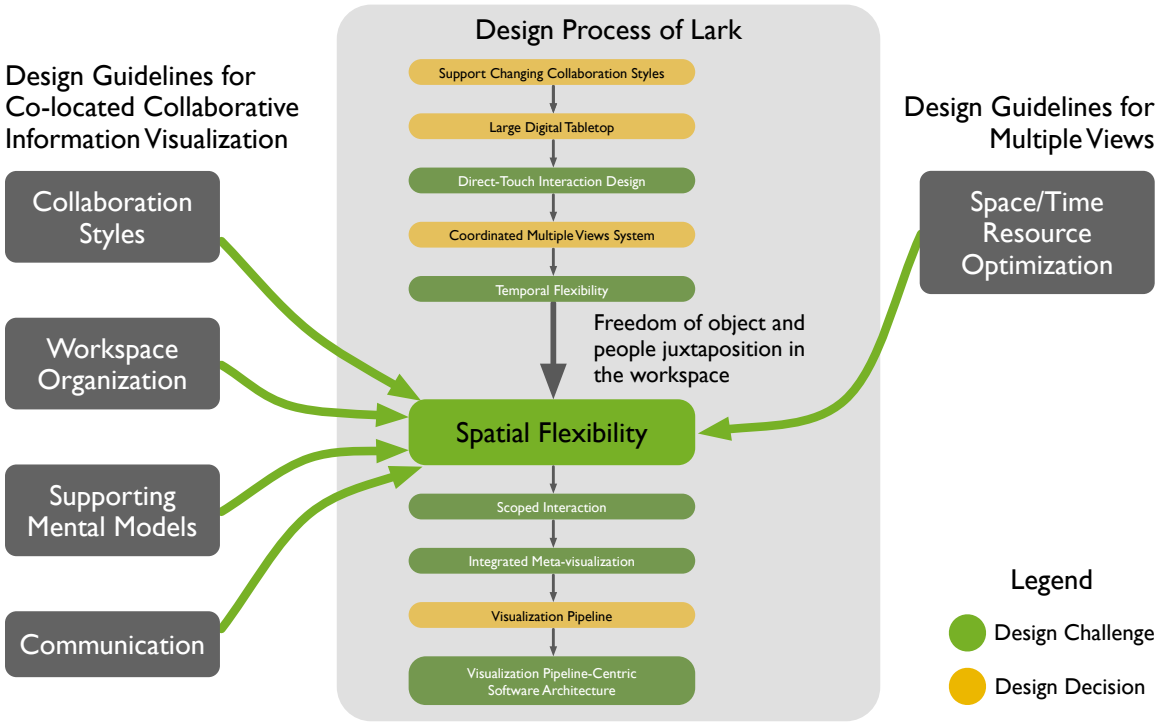


Figure 3.9: Lark's design challenge of *spatial flexibility*.

The spatial flexibility design challenge is supported by Isenberg and Carpendale [2007]'s guidelines for supporting mental models, workspace organization, and communication. It is also supported by Baldonado et al. [2000]'s guidelines for space/time resource optimization. The *supporting mental models* guideline (see Section 2.6.4.2) asserts that workspace items should have the ability to be individually placed, scaled, and organized throughout the workspace. Providing this functionality facilitates in the creation and maintenance of mental models by allowing group members to impose their own organizational principles to workspace items. Spatial flexibility also ties in with the *workspace organization* guideline (see Section 2.6.4.3). Unconstrained spatial arrangement of workspace items enables the emergence of an organic workspace organizational structure, allowing individual team members and the group at large to establish both individual and group work areas [Scott et al., 2004] in a dynamic fashion. The *communication* guideline (see Section 2.6.4.3) is supported too, as being able to spatially re-orient interface items has been shown to support ease of reading and collaborative communication practices [Kruger et al., 2005].

Lastly, the spatial flexibility design challenge also strives to adhere to the *rule of space/time re-*

source optimization for the effective use of multiple views (see Table 2.2). This guideline proposes that both the spatial and temporal costs associated with multiple views must be balanced. The former considers the amount of screen real estate occupied by multiple views, an important issue as screen real estate is a finite resource on any display and must be used effectively. The latter considers the temporal costs assumed by an analyst when he/she makes the context switch between views, as well as the computation time required to render alternative views. One step in balancing the spatial costs for multiple views is the design decision to use a large digital tabletop as Lark's hardware form factor (see Section 3.3.2). While the space of a large high-resolution tabletop display can still be easily exhausted, the display provides sufficient area to place multiple views next to one another for easy comparison. Empowering analysts with the ability to choose their own spatial arrangement of workspace objects further mitigates the spatial costs of multiple views, as analysts have full control over space usage within the workspace. Analysts can therefore optimize the usage of screen real estate to best support the task at hand, rather than conforming to a predefined layout. The spatial flexibility design challenge does not address the issues of the associated temporal costs with multiple views. This aspect is discussed later on in Section 3.3.8.

3.3.7 Scoped Interaction

Scoped interaction is another concept that my research addresses when supporting different collaboration styles. The research challenge of *how to provide scoped interaction* (see Section 1.5) also identifies this need. Providing a flexible definition of interaction scope is important in collaboration so that team workers can avoid interfering with each other's tasks when working individually and coordinate their interactions when working as a group. Returning to the four collaborative scenarios introduced earlier in this chapter (see Section 3.3.1), workspace objects in these scenarios range from a single shared instance to multiple individual instances, and the coordination of interactions on these objects is an important consideration. The different collaborative cohesion styles inherent in Scenarios A through D have different requirements in the type of interaction coordination. During parallel work phases (such as D), interactions by each group member should stay separate so that actions, such as filters or view changes, remain local and do not interfere with another person's

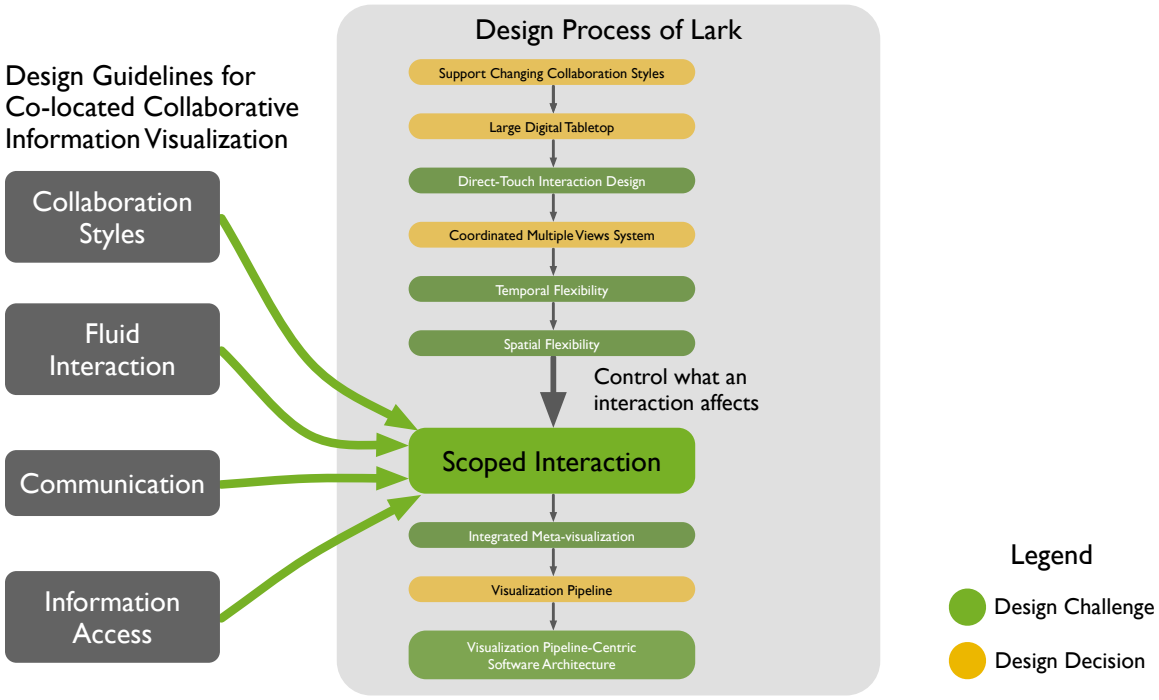


Figure 3.10: Lark's design challenge of *scoped interaction*.

current activity. During the transition towards Scenarios C and B, coordinating views between team members may benefit from the integration and relation of joint analysis results. Empowering collaborators with the ability to decide how their interactions should be coordinated would support these different types of collaboration styles, and the scoped interaction design challenge investigates how this kind of support can be provided. Moreover, the goal is to keep concurrent interaction individually scoped, allowing collaborators to choose how information will be linked and how changes to this information will be propagated, in addition to providing immediate and persistent information to all group members about interaction scoping.

The notion of providing explicit interaction scoping is supported by the design guidelines for collaboration styles, communication, information access, and fluid interaction from Isenberg and Carpendale [2007], all of which fall under the category of *coordination* within the collaborative environment (see Section 2.6.4.3). The *collaboration styles* guideline is directly followed as the primary objective of this design challenge is to provide a flexible definition of interaction scope, thereby supporting the coordination needs of different types of collaboration. The design challenge also speaks

to the *communication* guideline as an important part of interaction scoping is facilitating workspace awareness by making an action's interaction scope visually explicit within the collaborative work environment. Communicating the extent of an individual action's effect allows group members to coordinate their actions at an *information access* level as well. In this situation, information access does not necessarily need to be explicitly enforced, but rather the awareness provided by simply informing group members means that this information is no longer implicit and group members can now act on this knowledge. Furthermore, in adhering to the *fluid interaction* guideline, changing the interaction scope should be a simple and lightweight operation. Allowing the interaction scope to be easily changed, collaborators can then easily change their degree of collaborative cohesion, moving from individual work, where interaction is locally scoped, to group work with a more global interaction scoping.

3.3.8 Integrated Meta-visualization

An important part of realizing scoped interaction is informing collaborators of the extent to which different interactions are scoped. The next design challenge in the development of Lark looks at how this information can be effectively communicated. A promising method for providing visual awareness of how group member's actions relate is through an *integrated meta-visualization* [Weaver, 2005], as introduced in Section 2.8. Integrated meta-visualizations have been used previously in CMV environments to assist in the cognition of how multiple views are related to one another [Weaver, 2005; Collins and Carpendale, 2007]. Lark employs a similar principle, using a meta-visualization to illustrate the connections between the multiple views in the visualization environment, in addition to communicating the interaction scope of actions performed on these visualization views. To this end, my research gathers together the following series of design goals for creating an integrating meta-visualization:

- Create an integrated meta-visualization [Weaver, 2005] within the visualization workspace.
- Make the relationships between multiple views explicit (e. g., show that two visualizations are individual instances of the same underlying data set and to what extent).
- Bring to the surface the underlying coordination graph of coordinated multiple views.

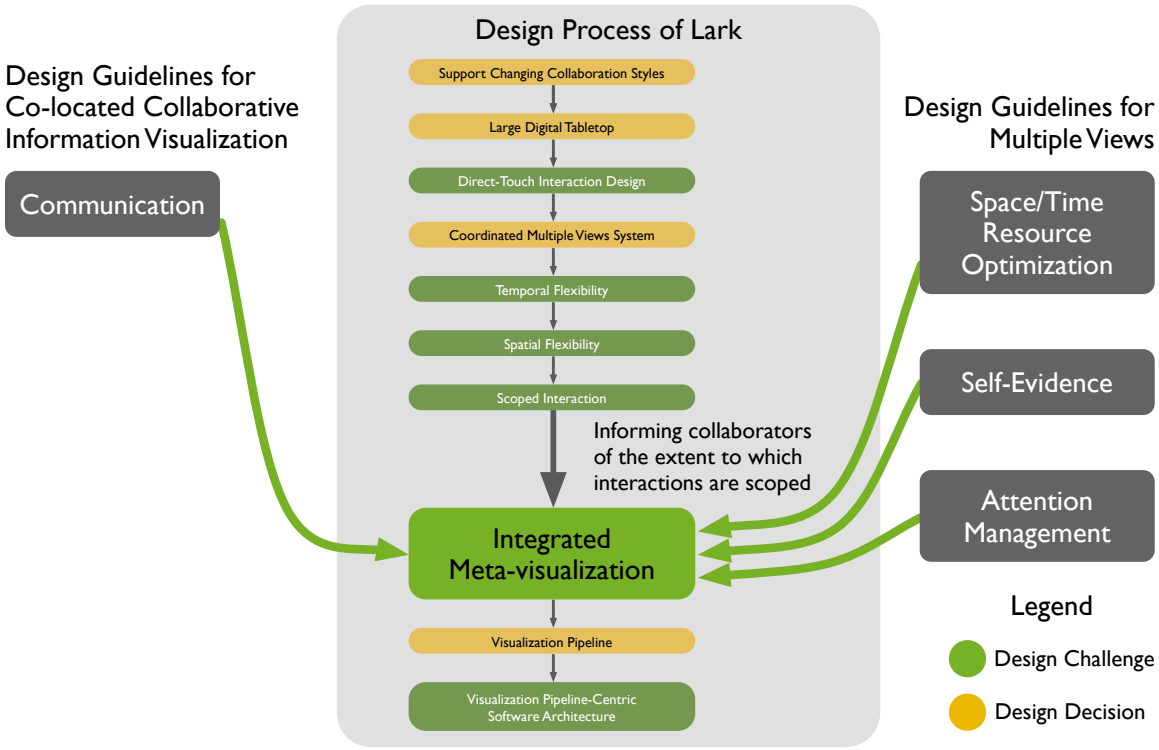


Figure 3.11: Lark's design challenge of creating a *integrated meta-visualization*.

- Represent propagating interactions (e. g., make it explicit that altering this view will affect these other two views as well).
- Clarify the distinction between value and view operations [Chi and Riedl, 1998] (as introduced in Section 2.4.1.3).
- Keep the visuals minimal.
- Embed necessary interactions within the meta-visuals.

The integrated meta-visualization design challenge makes use of the design guidelines of communication by Isenberg and Carpendale [2007] (see Section 2.6.4.3) and self-evidence, attention management, and space/time resource optimization by Baldonado et al. [2000] (see Table 2.2). Providing a meta-visualization within Lark's collaborative environment is strongly motivated by the considerations touched on in Isenberg and Carpendale's *communication* guideline, as the meta-visualization provides a visual means of communicating an action's interaction scope to collaborators. Related

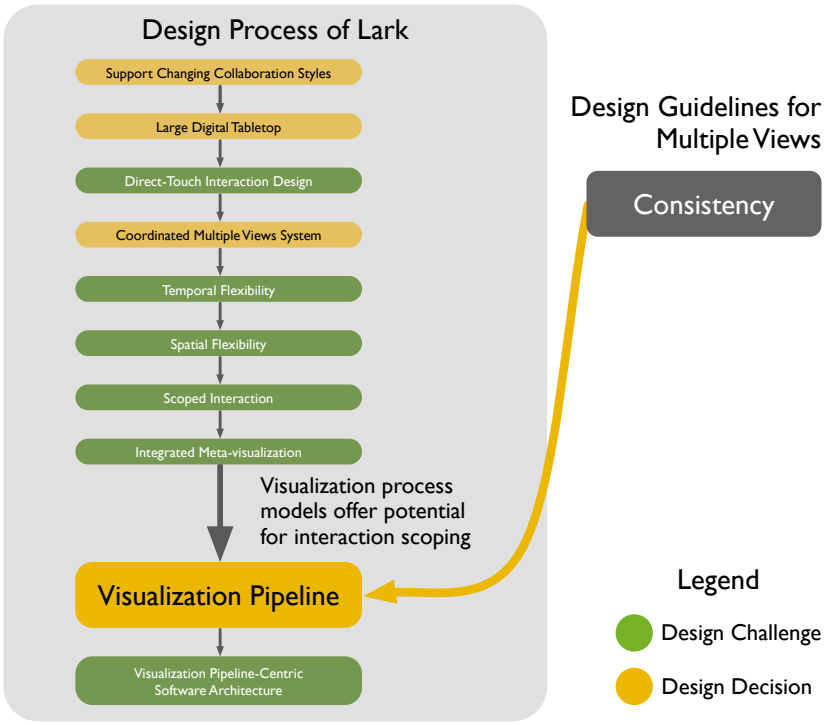


Figure 3.12: Lark's design decision of structuring the integrated meta-visualization after a *visualization pipeline*.

to the mechanics of collaboration [Gutwin and Greenberg, 2000], the meta-visualization provides consequential communication, coordination of action, monitoring, and protection. Furthermore, by surfacing the underlying coordination graph of Lark's CMV environment, the meta-visualization follows the *rule of self-evidence* as the relationships between multiple views are made visually explicit. The meta-visualization can also be used as a means of directing analysts' attention towards "the right view at the right time", in accordance with the *rule of attention management*. Lastly, the meta-visualization can also help in reducing the temporal costs associated with switching between multiple views by communicating contextual information such that analysts can quickly re-orientate themselves in the new view. This approach can help in balancing the temporal costs of multiple views, as suggested in the *rule of space/time resource optimization*.

3.3.9 Visualization Pipeline

The next design decision in Lark's design process is to structure the integrated meta-visualization after a *visualization process model* (discussed in detail in Section 2.2 through Section 2.4). Visual-

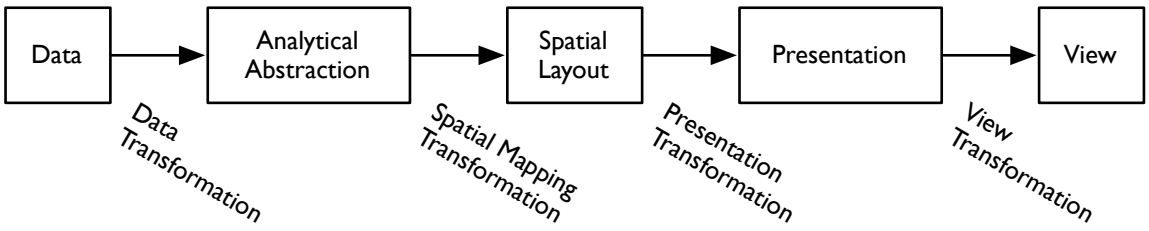


Figure 3.13: The visualization pipeline used in Lark (shown above) is similar to Carpendale [1999]'s pipeline, except that it uses different terminology for the pipeline states and transformations.

ization process models, or *visualization pipelines*, offer the potential for interaction scoping by deconstructing the visualization process, of transforming raw data into interactive computer graphics, into a series of chained data processing steps. In this model, any interaction with the end visualization is mapped to a modification of a discrete pipeline step. Altering the processing operation of a pipeline step thereby influences the data received by subsequent pipeline steps. Branching the visualization pipeline between pipeline steps enables the end visualization views to utilize common *upstream* pipeline steps, with unique pipeline steps occurring *downstream* from the branching point. These discrete pipeline steps and the rooted tree that connects the steps to one another provides an effective mechanism for *interaction scoping*, as the extent of an action's effect is logically captured within the process model.

Lark's visualization pipeline is illustrated in Figure 3.13. This state-based visualization process model is similar to the pipeline introduced by Carpendale [1999] (see Section 2.4.3), except that it uses slightly different terminology for the pipeline states and transformations, although the semantics of each remain consistent with Carpendale's definitions. The pipeline begins with the *data* state which contains raw unprocessed data. A *data transformation* processes this raw data into a form that is more readily consumable by subsequent steps in the visualization pipeline, resulting in the *analytical abstraction* state. The data transformation could include operations such as value filtering (see Section 2.4.1.3), statistical data analysis methods (e. g., cluster analysis), and deriving meta-data from the raw data. From the analytical abstraction state, a *spatial mapping transformation* applies a visual representation to the processed data, adding geometric information to the individual ele-

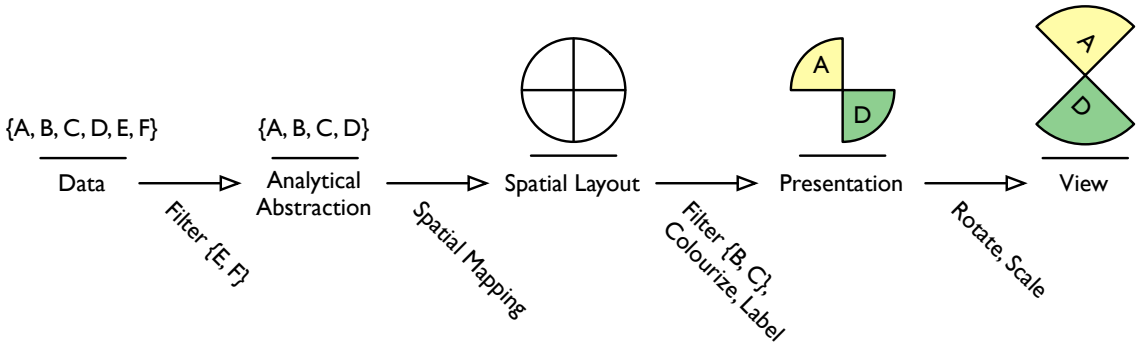


Figure 3.14: Lark's visualization pipeline illustrated using an example visualization process of visualizing set data.

ments. From this *spatial layout* state, a *presentation transformation* further processes the data's visual representation into the *presentation* state. This transformation includes operations such as colourization, labelling, highlighting, view filtering (see Section 2.4.1.3), and spatial re-organization (e. g., magnification). The final *view transformation* moves data from the presentation state to the *view* state. This transformation includes operations such as projection transformations (e. g., perspective and orthographic projections) and view operations (e. g., rotation, pan, zoom, scale, and shear).

Figure 3.14 illustrates an example visualization process where the individual operations are categorized according to Lark's visualization pipeline. In this example, the initial raw data is a set of elements, $\{A, B, C, D, E, F\}$. The data transformation performs a filtering operation, removing elements $\{E, F\}$ from the initial set, resulting in the set $\{A, B, C, D\}$ at the analytical abstraction state. Next, the spatial mapping transformation visually represents the four element set as a pie chart with four equally sized pieces. At the presentation transformation, set elements $\{B, C\}$ are filtered from the view, and the remaining elements are labelled and colourized with a qualitative colour scheme. Lastly, the view transformation rotates the visualization clockwise by 45° and scales the elements by 150%.

As mentioned above, Lark's visualization pipeline can also be branched at different pipeline states. This concept is illustrated in Figure 3.15 using the same initial set data from Figure 3.14, with different transformation operations and two branching points, resulting in three end visualization views. It is important to note that each subsequent step in the visualization pipeline builds upon the decisions made in the previous steps. Work by Heer and Agrawala [2008] on asynchronous remote

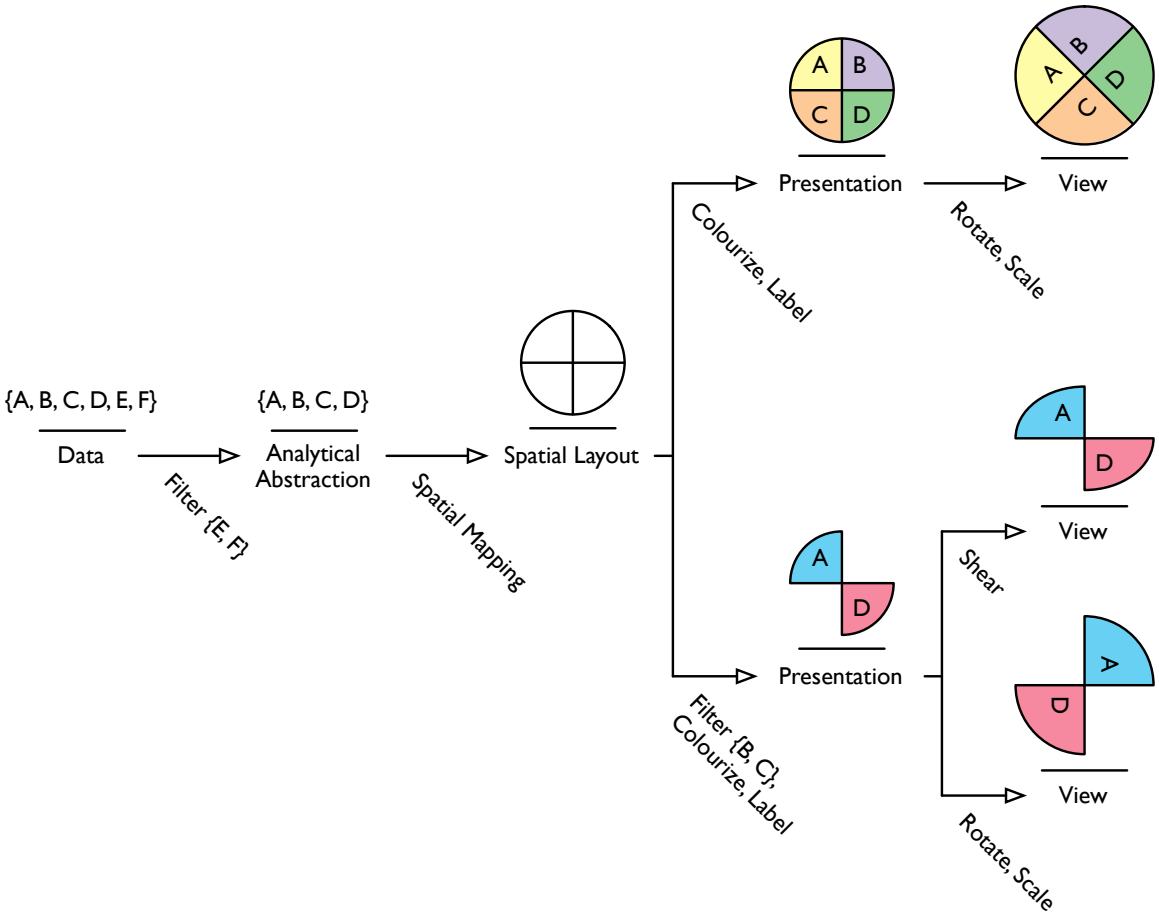


Figure 3.15: Branching Lark's visualization pipeline illustrated with three end visualization views of common underlying data.

collaboration suggested that these discrete stages in the visualization pipeline can be used to coordinate collaborative activities. Lark's pipeline contains three specific coordination points for possible interaction: analytical abstraction, spatial layout, and presentation states. These are essentially *collaboration coordination points (CCPs)* which arise from the pipeline concept, allowing the development of a *collaboration coordination tree* that specifies which views are linked and at which level of the pipeline. For instance, in Figure 3.15 any changes to the spatial layout state will affect all three subsequent views. Changes to the bottom presentation state will only affect the lower two view states, leaving the upper view unaffected.

The visualization pipeline thereby offers a mechanism for modelling interaction scoping. Multiple copies of shared visualizations can be provided by utilizing a common visualization pipeline

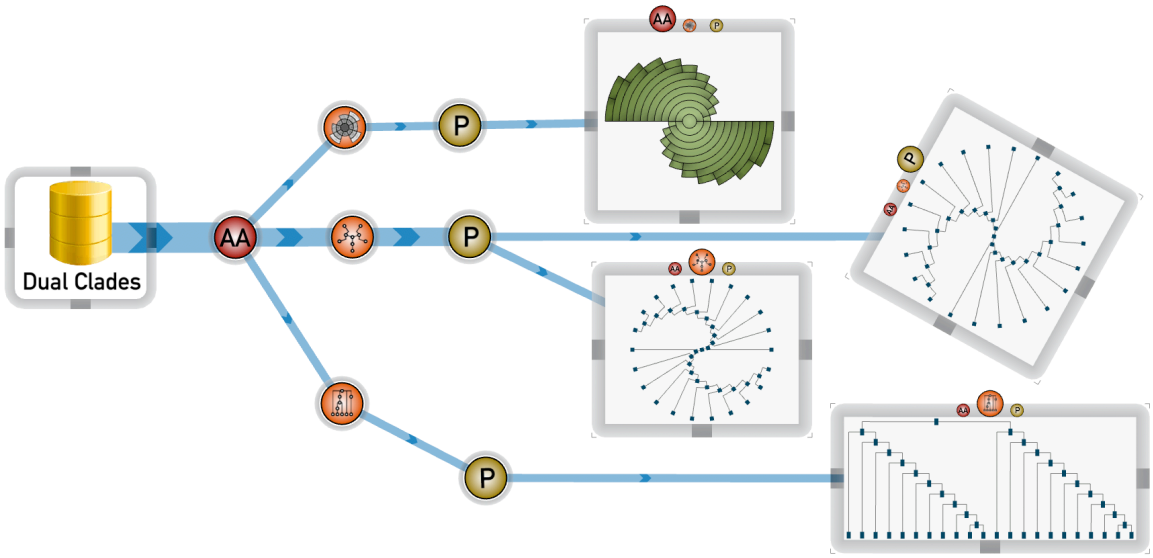


Figure 3.16: Lark's workspace showing four views of a common data set, "Dual Clades", linked together with an integrated meta-visualization making the underlying visualization pipeline visually explicit.

up to the presentation state and then branching into multiple view states (as shown in the bottom two views of Figure 3.15). These multiple views are "shared" in that any interaction with common visualization pipeline steps will influence both views, however they can be independently positioned within the workspace as each view has a unique view transformation (as seen in the separate sheer, and rotation and scaling operations in Figure 3.15). Multiple individual copies can be provided by utilizing multiple distinct visualization pipelines all branched off of the initial data state. Here, the visualizations are based on the same data set, yet have the potential for completely unique end views. Moving the branching point connecting two views further down the pipeline increases the amount of cohesion between the views, providing a flexible means of defining interaction scope. Furthermore, with this pipeline approach, each view has its own unique view transformation, and all views (shared or individual) can be freely positioned about the workspace. This ability realizes the design challenge of spatial flexibility of objects within the workspace.

Lark's visualization pipeline is made visually explicit within the collaborative workspace through the meta-visualization. Figure 3.16 presents an image of Lark's workspace where the connections within the visualization pipeline linking four visualization views are visually represented by the meta-

visualization. A full explanation of the meta-visualization's visual encoding is discussed in detail in Section 4.1.2 of the following chapter.

In summary, the design decision to coordinate views within Lark's visualization workspace around the visualization pipeline facilitates the realization of the design challenges of scoped interaction and spatial flexibility. This approach also addresses the coordination mechanism of Lark's multiple view environment, as this coordination is directly modelled using the visualization pipeline. Making the visualization pipeline visually explicit within Lark's workspace structures the integrated meta-visualization. This design decision is supported by the *rule of consistency* for multiple view systems [Baldonado et al., 2000]. Lark's visualization pipeline provides a well defined structure for modelling the coordination between multiple views, as each pipeline path is always made up of five discrete states, regardless of the transformation operators at work. Since the pipeline is presented visually within Lark's workspace, this leads to a consistent interface illuminating the coordination structure of Lark's multiple views.

3.3.10 Visualization Pipeline-Centric Software Architecture

The final design challenge in the development of Lark is engineering Lark's software architecture as *visualization pipeline-centric*. That is to say, the clear division between pipeline states and the transformations between states, as illustrated in Figure 3.13, should be distinctly manifest within Lark's software architecture. Structuring the software architecture after the visualization pipeline has been identified as one of the pragmatic utilities of these conceptual process models [Duke et al., 2006]. Lark's implementation features all the interaction design, user interface, and information visualization concepts introduced in the design decisions and challenges mentioned above. Furthermore, the implementation must ensure that the system performs at interactive rates. This has been identified in the *system setup* guideline by Isenberg and Carpendale [2007] as an important issue for collaborative information visualization systems, as a system performing below interactive rates profoundly impedes overall usability.

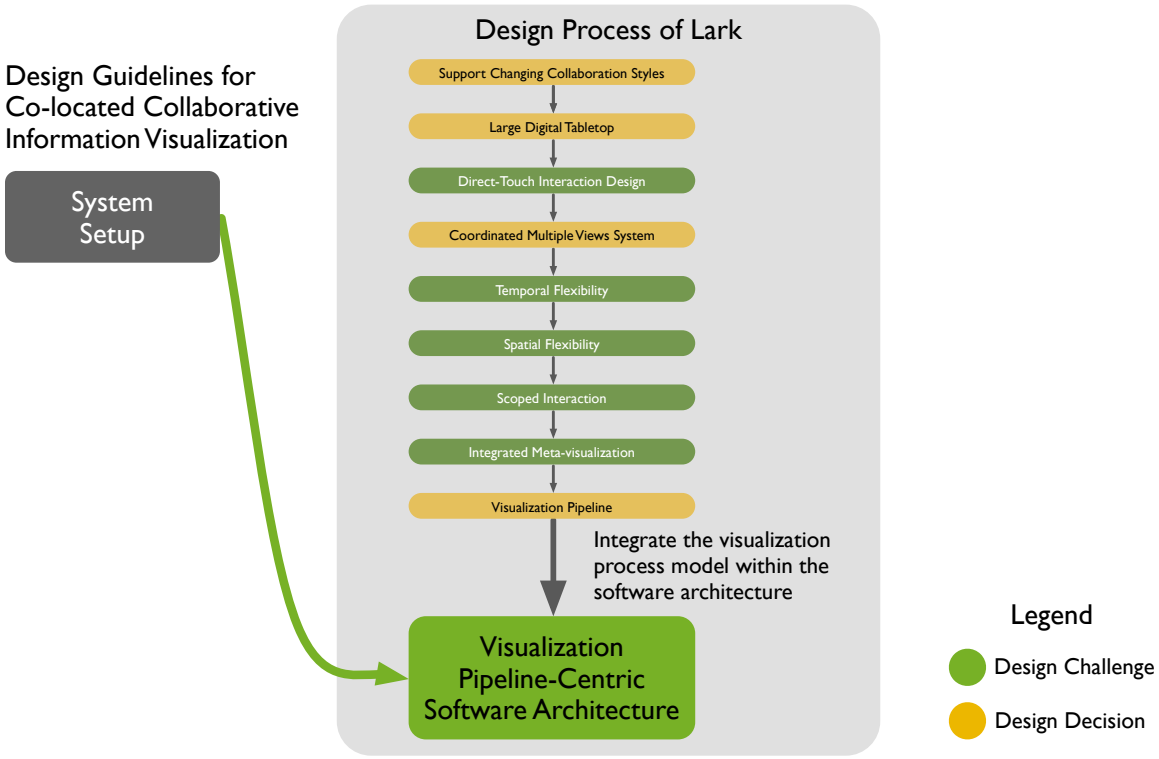


Figure 3.17: Lark's design challenge of engineering Lark with a *visualization pipeline-centric software architecture*.

3.4 Discussion

Lark's design process introduced in the previous section adheres to many of the design guidelines from Isenberg and Carpendale [2007], and Baldonado et al. [2000], however not all of them are followed. In the guidelines for co-located collaborative information visualization systems [Isenberg and Carpendale, 2007], the two guidelines that where not utilized are *task history* and *perception* (see Figure 3.1). While these guidelines represent promising areas for future research (some of which have already been pursed [Wigdor et al., 2007]), they are beyond the research scope of this thesis.

The concept of coordinating collaboration around the information visualization pipeline has been explored in previous research. Work on asynchronous remote collaboration by Heer and Agrawala [2008] has identified the idea of using the information visualization reference model as "entry points for collaborative activity" [Heer and Agrawala, 2008, p. 51], as mentioned previously. In earlier work, Wood et al. [1995] used the idea of coordinating collaboration around the visualization pipeline in a

distributed system. This work is conceptually close to Lark, however the implementation was limited to an architectural design whereas my approach focuses on making different coordination points in the pipeline visually accessible and understandable.

Lark's coordination and collaboration concept is generalizable, and can be applied to the visualization of many different types of data. The prototype application introduced in Chapter 4 realizes the coordination and collaboration concept, focusing of visualizing a specific type of data: hierarchical data. Lark uses this particular data type as a compelling exemplar for visualization, however once again, Lark's concept is generalizable to other types of data.

3.5 Summary

This chapter introduced Lark's collaboration and coordination concept. Beginning with the background of Lark's design process in Section 3.1, an overview of the Lark system was presented in Section 3.2. This is followed by Section 3.3 introducing the system's conceptual design. This discussion was structured around Lark's design process, illustrated in Figure 3.1, which identified the design challenges and design decisions made in the research and development of Lark. Lark's design process begins with focusing the research effort on *supporting changing collaboration styles*. All other design choices are subordinate to this focus, identified as a design decision, albeit how to provide this type of support is an open and difficult question. Section 3.3.1 further characterizes mixed-focus collaborative work (the type of collaboration that is the focus of this research) and draws on potential strategies from the design guidelines by Isenberg and Carpendale [2007] on how support could be provided for changing collaboration styles. The next step in Lark's design process is the design decision to use a *large digital tabletop* as the hardware form factor used in this research. This decision is supported by previous research on mixed-focus collaboration [Rogers and Lindley, 2004; Tang et al., 2006] which identified large digital tabletops as a promising form factor for supporting synchronous co-located collaborative work. Section 3.3.2 presents the technical details of this hardware setup. The decision to use a touch-sensitive horizontal display has direct implications on the interaction design of the system. Therefore, what follows from the previous design decision is the challenge of creating a user interface specifically for direct-touch interaction, which is necessary to fully realize the poten-

tial of the tabletop form factor (Section 3.3.3). The next design decision seeks to address the support of changing collaboration styles through the use of a CMV environment within the visualization workspace (Section 3.3.4). This approach is supported by both guidelines for co-located collaborative information visualization [Isenberg and Carpendale, 2007], as well as, guidelines for *when to use* multiple views [Baldonado et al., 2000]. These guidelines compliment each other and suggest that providing multiple visualizations that are linked to one another via an interactive dependency is a promising technique for supporting collaboration.

The next design challenge in the development of Lark is creating a visualization environment that does not require a specific temporal ordering of actions (Section 3.3.5). Previous research [Isenberg et al., 2008] has shown that small groups involved in mixed-focus collaboration do not follow a strict temporal sequence of analysis tasks; rather, the analysis patterns are dynamic and spontaneous. Therefore, Lark seeks to support this type of behaviour by providing *temporal flexibility* of actions within the collaborative workspace. Another important characteristic of mixed-focus collaboration is that changes in collaborative cohesion in large workspaces is commonly accompanied by changing team member locations and the associated reorganization of workspace items [Scott et al., 2003a]. This characteristic is the motivation behind the *spatial flexibility* design challenge which strives to provide support for the free movement of objects in and people around the collaborative workspace (Section 3.3.6). The next design challenge is *scoped interaction*, introduced in Section 3.3.7. This challenge identifies that need for explicit control over the extent of an interaction's effects, such that group members can avoid interfering with each other's tasks when working individually and coordinate their interactions when working as a group. This requires a flexible definition of interaction scope which can be dynamically set by collaborators to best suit the immediate task at hand.

The next item in Lark's design process is determining a means of communicating to collaborators the extend to which different interactions are scoped. Lark uses an *integrated meta-visualization* to present this information, discussed in Section 3.3.8. The meta-visualization is also used to surface the underlying CMV coordination graph. This step in the design process is labelled as a design challenge since how to visually structure this information within the meta-visualization is an open question. The next design decision is to structure the meta-visualization after a visualization process

model, similar to the pipeline model proposed by Carpendale [1999]. Lark's visualization pipeline is shown in Figure 3.13. Visualization process models offer the potential for interaction scoping by deconstructing the visualization of data into a series of chained processing steps, and these discrete steps can be used to define an interaction's scope. The steps within a visualization process model also lend themselves towards well defined coordination points for collaborative activities, as identified by Heer and Agrawala [2008]. Lark terms these as CCPs, or *collaboration coordination points*, allowing the development of a *collaboration coordination tree* that specifies which views are linked and at which level of the pipeline. This coordination tree provides the model for structuring the CMV coordination graph. This *visualization pipeline* design decision discusion is found in Section 3.3.9. The final item in Lark's design process is the design challenge of architecting Lark's software after the visualization pipeline (Section 3.3.10). The implementation of Lark features all the interaction design, user interface, and information visualization concepts identified in the design process, with the whole system performing at interactive rates.

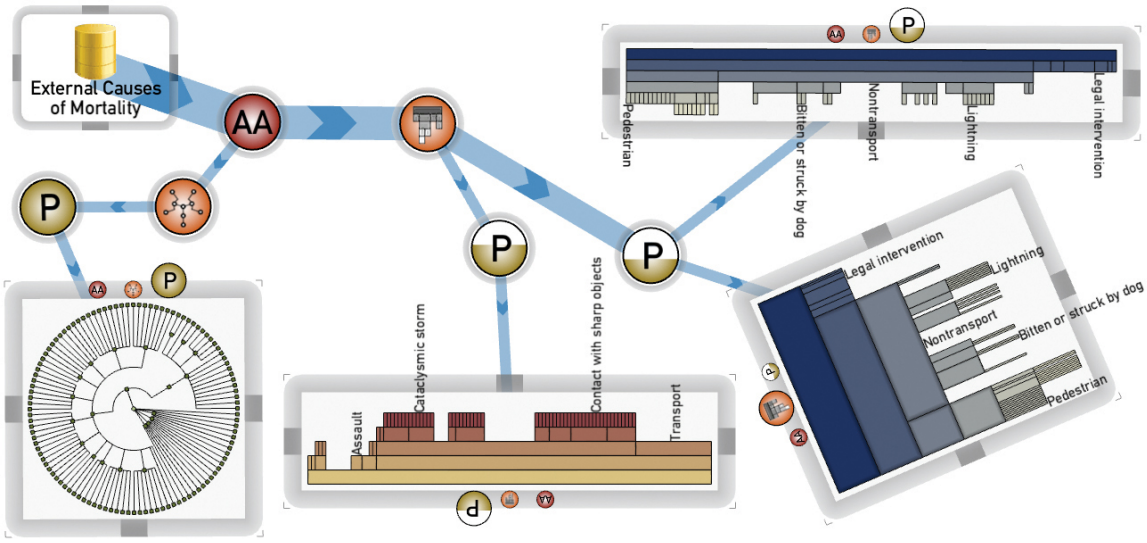
Building on this introduction of Lark's coordination and collaboration concept, the next two chapters present the interface and software architecture designs which realize these concepts. Chapter 4 discusses Lark from the perspective of someone using the system, identifying how the system's interface follows Lark's design decisions and manifests specific approaches to the different design challenges. Chapter 5 introduces Lark's software architecture from a development perspective. These next two chapters segment Lark's design process in the following manner. Chapter 4 touches on the first nine design challenges/decisions, Section 3.3.1 through Section 3.3.10. Chapter 5 addresses the last item in the design process, Lark's software engineering design challenge from Section 3.3.10.

Chapter 4

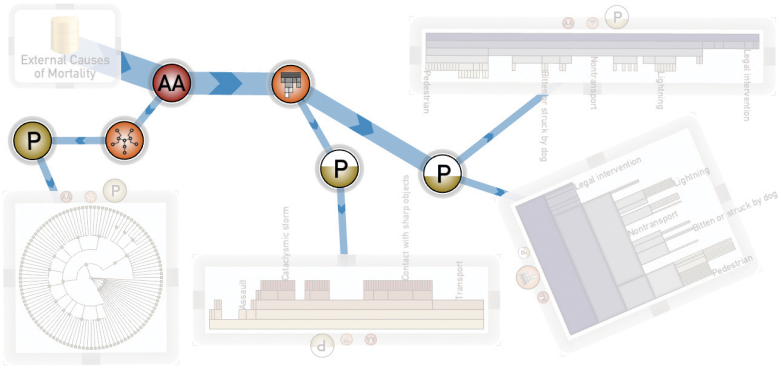
Interacting with Lark

In this chapter I present the interface design of Lark, which realizes Lark's collaboration and coordination concept as introduced in the previous chapter. This discussion presents Lark as seen from the perspective of someone using the system, and looks at the particular design decisions that were made in creating the visual and interaction aesthetic of Lark. The visual design of the collaborative visualization environment is presented, along with the system's interaction design capturing what it is like to work within this environment.

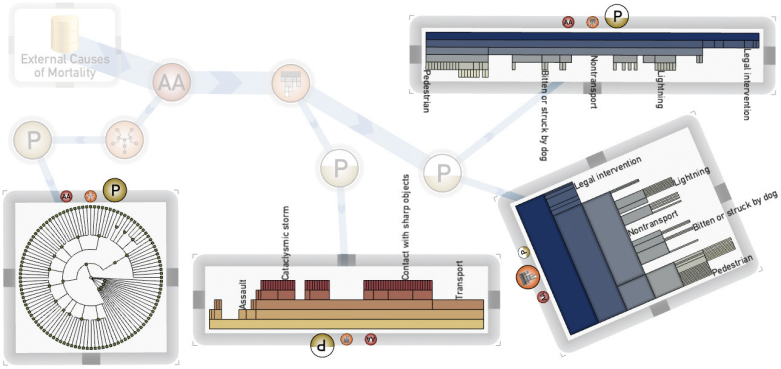
This chapter is organized as follows. Section 4.1 presents an overview of Lark's information visualization environment, introducing the visual design of the meta-visualization and what it represents. Section 4.2 presents the interaction techniques for creating new views and coordinations between views within Lark's coordinated multiple views (CMV) system. Section 4.3 covers more of Lark's interactions techniques, with particular attention paid to how the interaction design realizes Lark's design challenges, as identified in Chapter 3. Lastly, Section 4.4 explains how Lark's visualization pipeline, represented by the meta-visualization, can be duplicated or *cloned*, and why this is an important operation for collaboration support. Section 4.5 concludes this chapter with a summary discussion.



(a) Lark's collaborative visualization environment.



(b) The pipeline meta-visualization.



(c) The individual views.

Figure 4.1: Lark's collaborative visualization environment: single data set "External Causes of Mortality", four coordinated views, plus a meta-visualization of the visualization pipeline which explicitly shows how the four views are linked to one another in the CMV system. The original image—Figure 4.1(a)—has been altered for illustrative purposes in Figure 4.1(b) and Figure 4.1(c), focusing on the meta-visualization and the individual views.

4.1 Lark's Information Visualization Environment

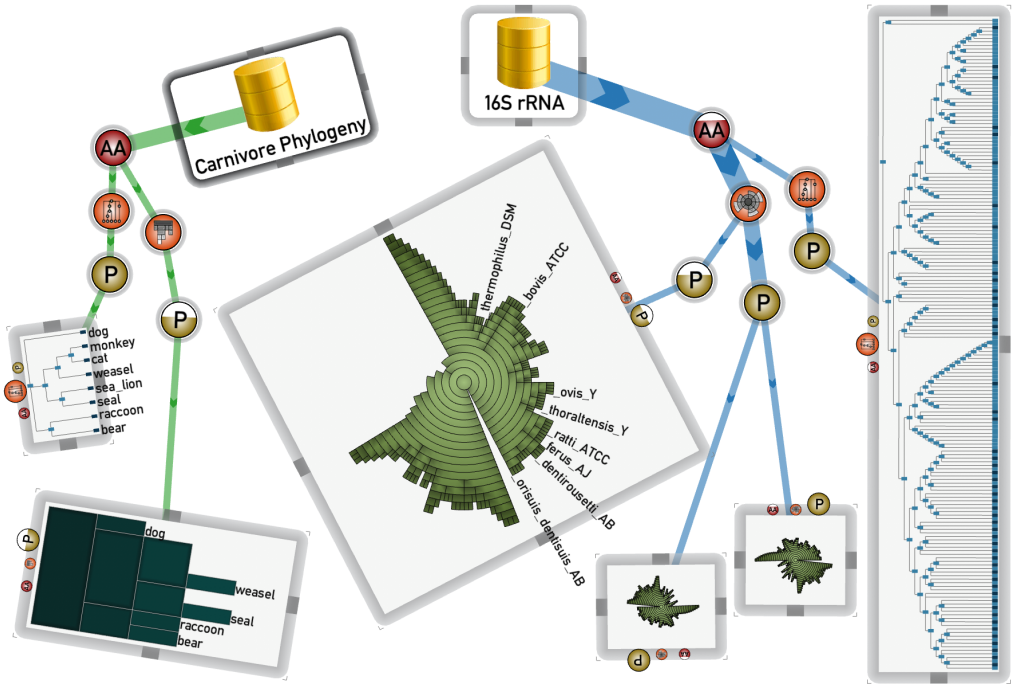
Lark is an information visualization system where an integrated meta-visualization shows the links and relationships among multiple coordinated views. Figure 4.1 presents an example of Lark's interface: a single data set, four coordinated views, and a linking meta-visualization illustrating the relationships between the data set and the views.

Note that unless otherwise stated, all images of Lark's interface come directly from the system, demonstrating what Lark looks like from the view of someone using the system. The visualization workspace has been organized so that the meta-visualization and the views are clearly visible for illustrative purposes. Furthermore, two distinct colour palettes for Lark's visualization workspace were designed—one for usage on rear projected tabletop displays and one for print graphics—and these colour palettes have been optimized for their respective presentation mediums. The two colour palettes are briefly summarized as follows: the tabletop palette uses a neutral grey workspace background to avoid high contrast between foreground interface elements; the rest of the palette is designed around this neutral tone, with localized high contrast regions where appropriate (e. g., white text for labelling individual elements in the hierarchical data). The print graphics palette, on the other hand, uses a white workspace background with the rest of the palette subordinate to this colour. Both palettes are shown in Figure 4.2.

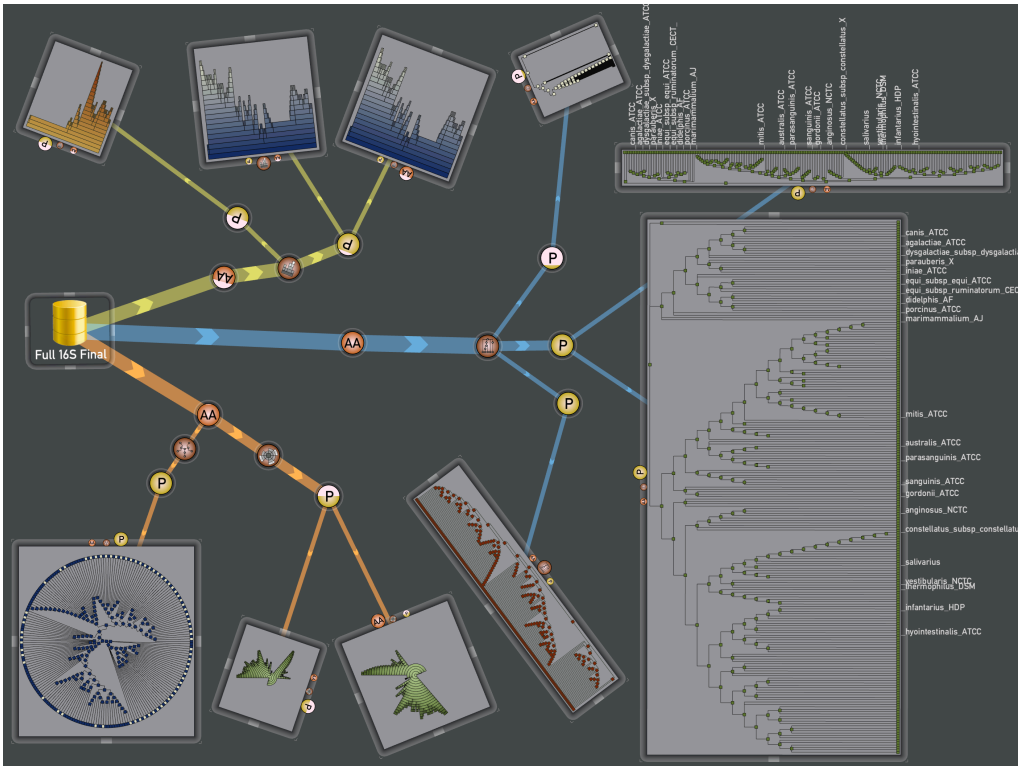
4.1.1 View Representation

Lark's visualizations of hierarchical data are contained within individual view-panes, which are mobile and resizable, and can be placed anywhere in the workspace. From here on, these are simply referred to as *view-panes* within Lark. The ability to move and resize the view-panes realizes the design challenge of *spatial flexibility*, as introduced in Section 3.3.6. Allowing visualizations to be freely moved about the workspace enables group members to rearrange views to best support the current task and collaboration style.

The visualizations contained in the view-panes are the focus of the teamwork that is to be supported by Lark. Figure 4.3 shows individual view-panes visualizing a common hierarchical data set with different tree layout algorithms. The view-panes are framed with a semi-transparent grey



(a) For print graphics.



(b) For tabletop displays.

Figure 4.2: Examples of the two colour palettes designed and optimized for two different presentation mediums: print graphics and rear projected tabletop displays.

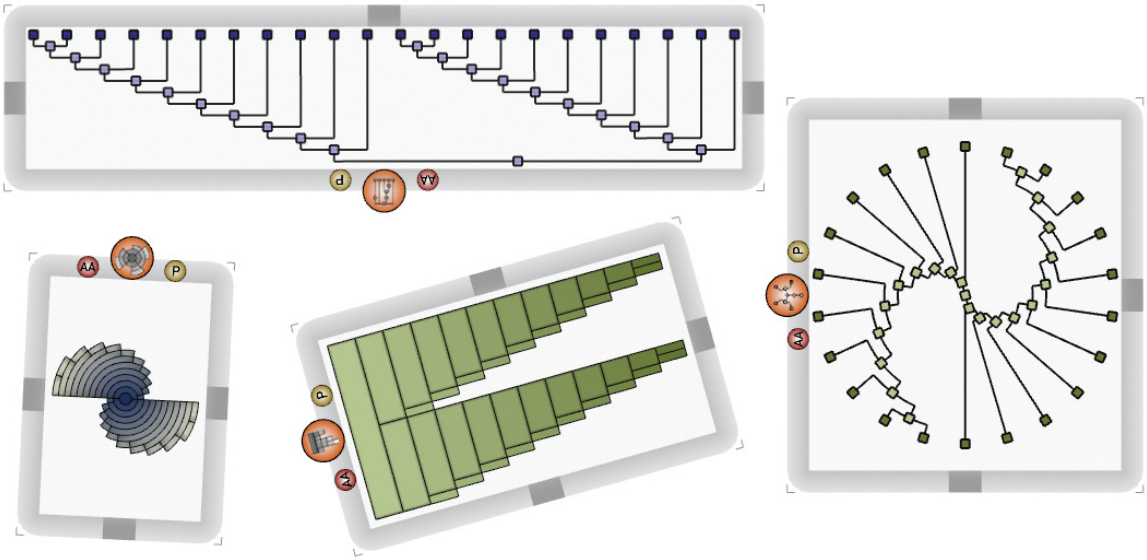


Figure 4.3: Individual view-panes, visualizing the same data set with different tree layouts. All view-panes are mobile and resizable through direction interaction with the surrounding grey border. In this image, the meta-visualization has been omitted for clarity.

border, which provides resizing, translation, and integrated rotation and translation (RNT) [Kruger et al., 2005] operations. The border's width is set to allow for direct-touch interaction, which speaks to the design challenge of designing for the touch-sensitive tabletop display form factor (see Section 3.3.3). Resizing is initiated from any of the four corners, translation from the darker grey regions of the border, and RNT throughout the rest of the frame. The three icons at the top of the frame are radio buttons, with the current selection indicated as the larger of the three. Data interactions are supported through gestures made directly on the data visualization. Through interactions with the radio buttons, data interactions are scoped and linked; this is discussed in more detail in Section 4.3.

4.1.2 Meta-visualization

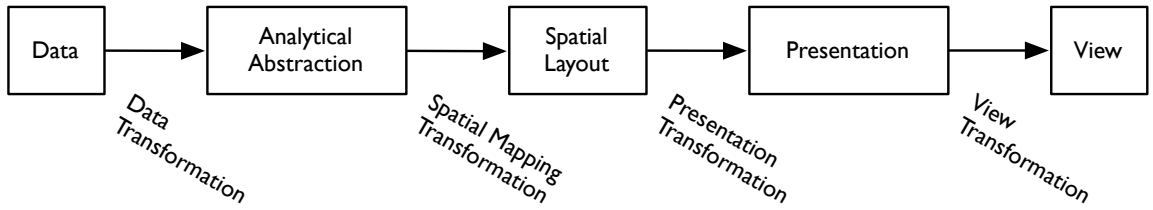
Lark's underlying visualization pipeline, as introduced in Section 3.3.9, is explicitly represented within the workspace via an integrated meta-visualization. The conceptual pipeline and two examples of its visual representation are shown in Figure 4.4, illustrating visualization pipelines of varying levels of complexity. Each visualization pipeline contains a data set view-pane, three types of collaboration coordination points (CCPs) for possible interaction—analytical abstraction, spatial layout, and presentation—and ending in a final view. These pipeline components are connected to one another

with coloured edges; connectedness has been shown to be a more effective grouping principle over techniques such as colour, proximity, shape, or size [Palmer and Rock, 1994]. Each new visualization pipeline branching off a data set has a unique edge colour, as illustrated in Figure 4.2(b). The relatively more complex visualization pipeline in Figure 4.4(c) contains numerous views of a single data set, where the connecting views form a *collaboration coordination tree* (as introduced in Section 3.3.9).

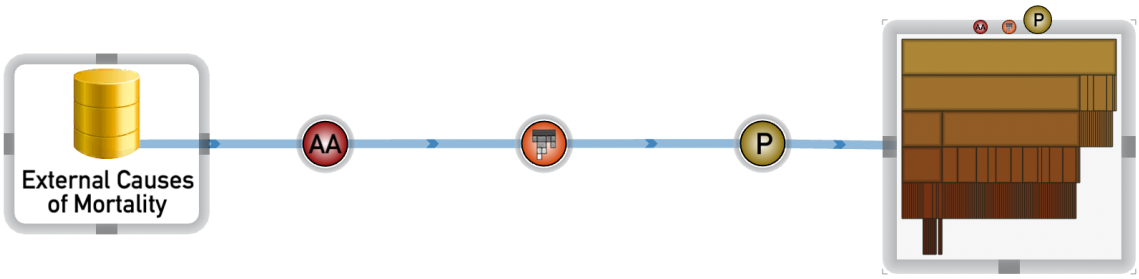
The data set view-pane is composed of a data icon and a text label, framed with a semi-transparent grey border. The borders of the data set and the visualization view-panes present a consistent look and feel of the interface, offering the same spatial arrangement operations, accessed via the same interaction techniques. Pane translation is initiated from any of the dark grey regions located along the four cardinal points around the frame. Interaction with the rest of the light grey border initiates RNT. The resizing operator has been omitted from the data set view-pane as its utility was not sufficiently compelling.

The CCPs arise from the pipeline concept and are made visually explicit in the workspace, allowing the development of a collaboration coordination tree that specifies which views are linked and at which level of the pipeline. In the meta-visualization, the number of views that are linked at any point in the collaboration coordination tree are indicated by the thickness of the tree edge, as seen in Figure 4.4(c). The CCPs themselves are indicated by a circle and labeled with an icon which declares the relation to the pipeline. The icons are surrounded with a circular semi-transparent grey border, which provides RNT such that the CCPs can be fluidly repositioned throughout the workspace. The first pipeline state is the *analytical abstraction*, labeled "AA" in the icon, which contains fundamental data operations such as removal of non-pertinent aspects of the data. The next state takes the data in its "ready-to-examine-state" and creates a *spatial layout*. In Lark, a small icon indicates this state. The third state is *presentation*, labeled "P", which includes all temporary visual transformations, such as colour transformations and the addition of labels. The final state is the *view* state, which contains the summation of the choices made in the previous states, the ability to interact with the view, and the ability to indicate which pipeline phase the interaction should affect via the bordering radio buttons.

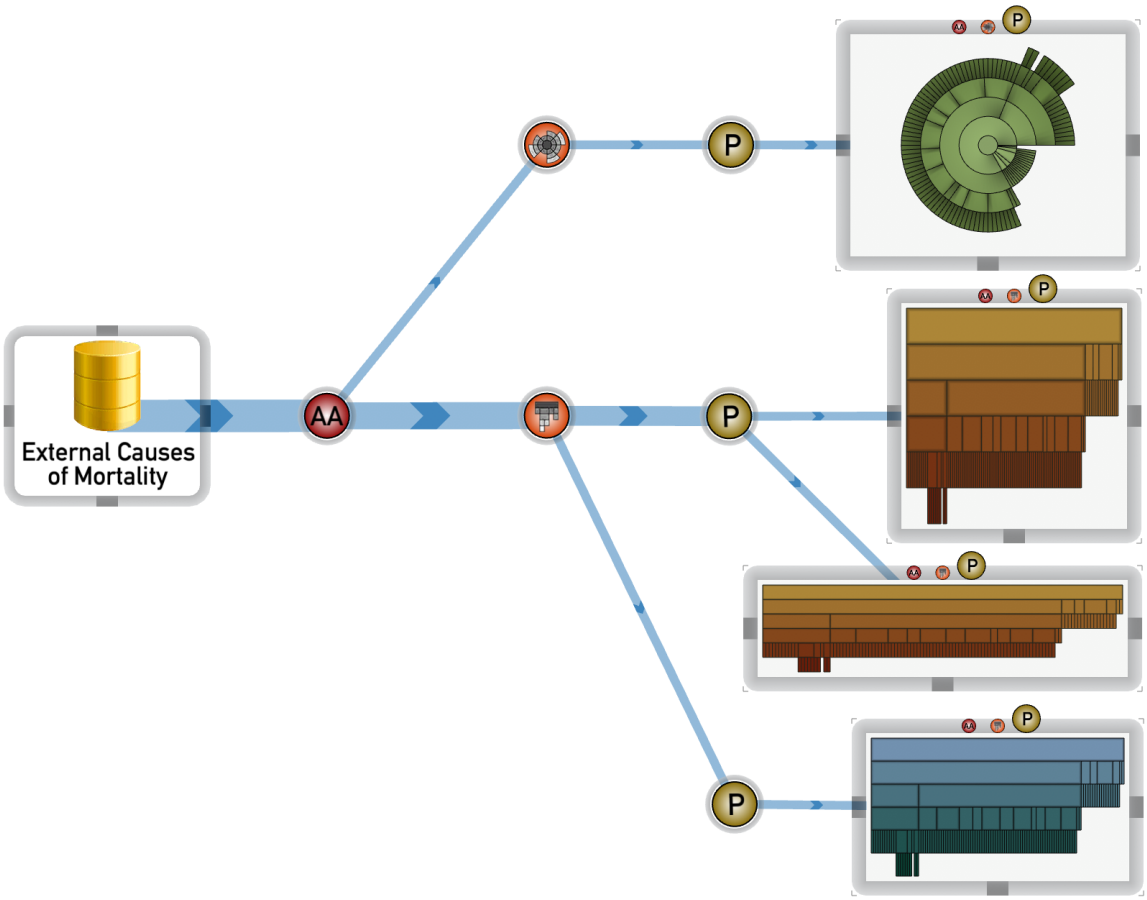
Each element of Lark's visualization environment—from data set view-panes, CCPs, and visualization view-panes—can be freely repositioned about the workspace, thereby addressing the de-



(a) Lark's conceptual visualization pipeline, as introduced in Section 3.3.9.



(b) A simple example of Lark's meta-visualization, explicitly representing the visualization pipeline.



(c) A more complex example of Lark's meta-visualization.

Figure 4.4: Lark's conceptual visualization pipeline and two example meta-visualizations demonstrating how the pipeline is visually represented within the visualization workspace.

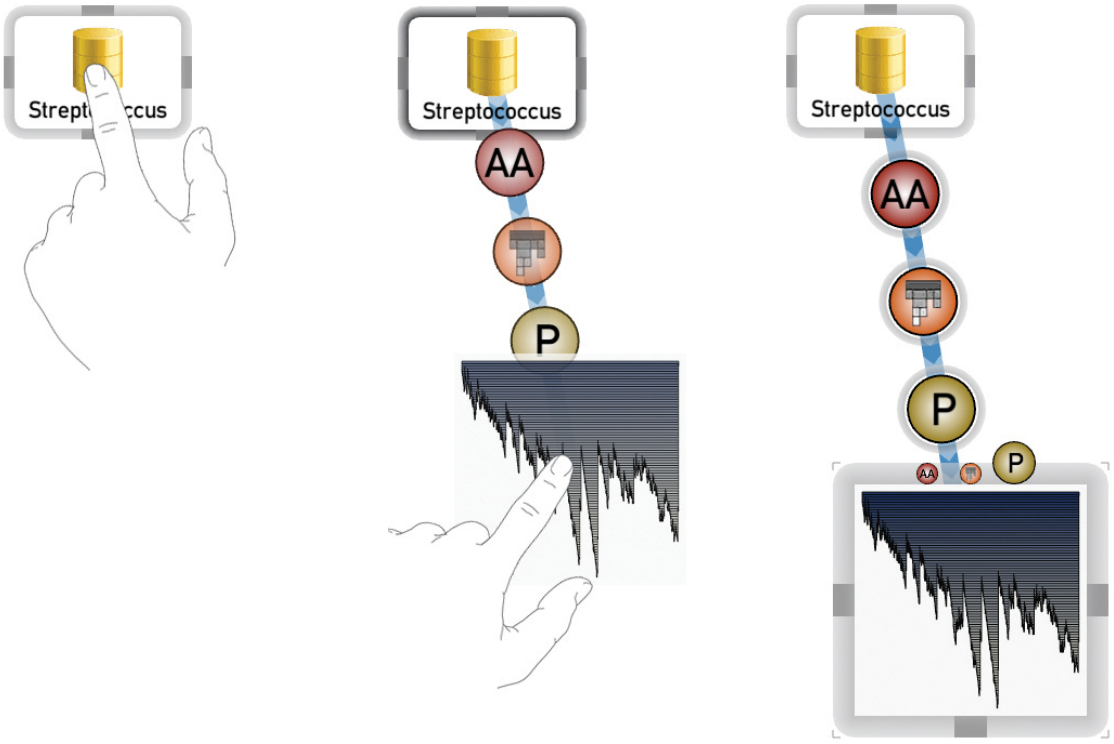
sign challenge of *spatial flexibility* (see Section 3.3.6). Collaborators can organize their workspace in which ever way they feel is most appropriate for the task at hand. When changing from different levels of collaborative cohesion, workspace objects can be repositioned to reflect changing work styles. These spatial operations use a consistent set of both interaction techniques and visual encodings for the available operations, presenting a unified look and feel to the interface. Lark's visual interface also realizes the design challenge of providing a *integrated meta-visualization* (Section 3.3.8) structured after the *visualization pipeline* (Section 3.3.9). The relationships between the multiple views are made *self-evident* [Baldonado et al., 2000] with the meta-visualization, following in a similar direction as work by Weaver [2005] on meta-visualizations in Improvise.

4.2 Visual Collaboration Coordination

Lark organizes and provides interaction with multiple visualizations of multiple data sets by illustrating the structure of the underlying visualization pipeline that was used to generate these visualizations. By making the visualization pipeline explicit, the relationships between individual views of a data set are emphasized. The visual structure of these relationships also provides awareness information to collaborators and shows how their interactions relate to one another. This section covers how views in Lark's CMV environment are created and coordinated with one another.

4.2.1 View Generation

The system starts by showing available data sources, each labelled and in its own data set view-pane. Creating a view of the data uses a *touch-drag-release* technique, illustrated sequentially in Figure 4.5. *Touching* the data source and then *dragging* away generates a semi-transparent view with default parameters for factors such as colour scales and layout type. The newly created view gains opacity as it travels with the touch-point away from the data source, to be *released* in its intended work location. Note in Figure 4.5(b), this new view is linked to its data source via a meta-visualization of the pipeline with CCPs at the analytical abstraction ("AA"), spatial layout (i. e., visual representation), and presentation ("P") states. Note also that the visualization view-pane's three interaction scoping buttons are initialized to the default "P", which keeps interaction local. This is a first branch of the underlying visualization pipeline which visualizes how views are coordinated (Figure 4.5(c)).



(a) Data source is touched.

(b) Dragging the finger reveals view and pipeline meta-visualization.

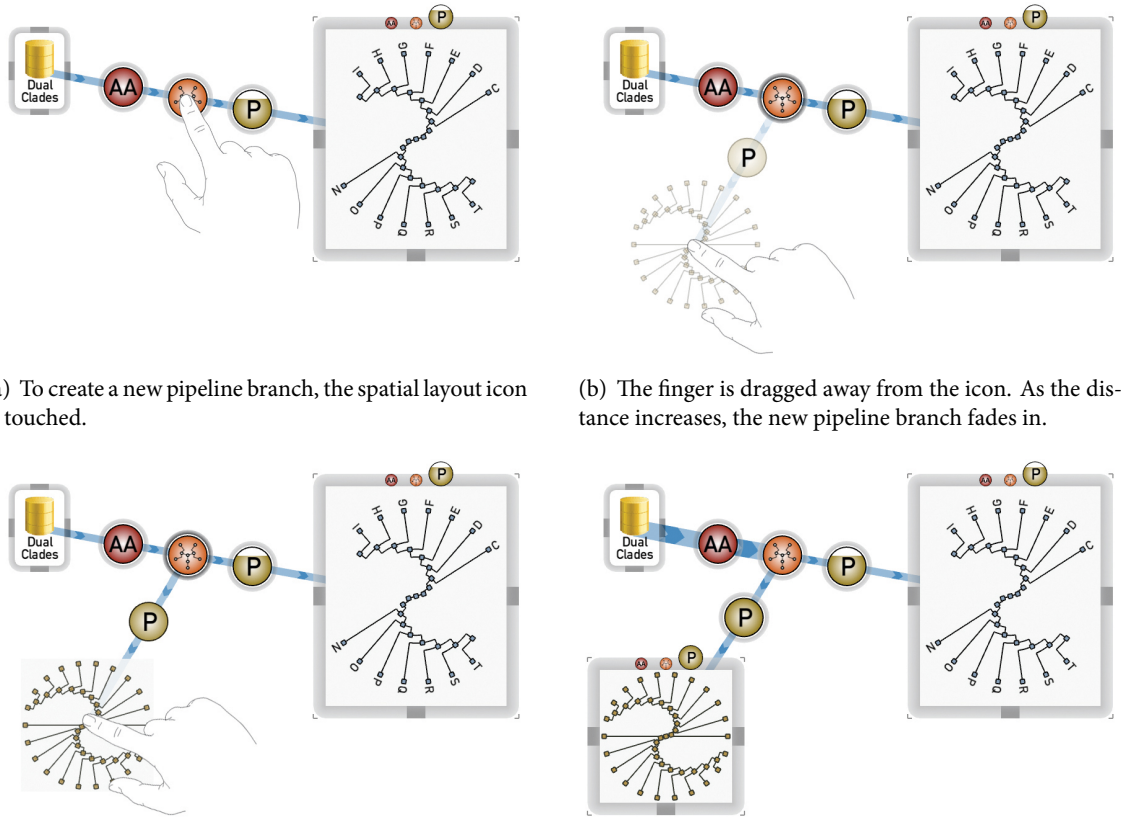
(c) Releasing reveals view-pane and scoping buttons.

Figure 4.5: The creation of a new view from a data source using a *touch-drag-release* interaction technique. The silhouette of a hand has been superimposed on the image for illustrative purposes.

4.2.2 Pipeline Creation and Branching

The visual representation of the pipeline in Lark is shown in Figure 4.1. Here the pipeline is structured as a *tree*, where the root is the initial data state, leaves are individual view states, and all other vertices are CCPs. All leaves have an equal depth of four dictated by the discrete states in the visualization pipeline.

Collaborators can dynamically create new visualization pipelines and branches from existing pipeline states through a *touch-drag-release* interaction technique. This interaction technique is illustrated in Figure 4.6. A new branch of the pipeline tree can be created from any CCP. To create a new pipeline branch, the CCP icon is *touched* and the touch point is *dragged* to the intended view-pane location. As the distance between the CCP and touch point increases, the newly created branch



(a) To create a new pipeline branch, the spatial layout icon is touched.

(b) The finger is dragged away from the icon. As the distance increases, the new pipeline branch fades in.

(c) The finger is released at the intended view-pane location.

(d) On release, the creation of the new branch is confirmed and interface decorators are added to the newly created components.

Figure 4.6: Interaction technique for creating a new pipeline branch off an existing spatial layout CCP.

proportionally gains opacity and is alpha blended into the workplace. *Releasing* the touch anywhere outside of the CCP icon boundary confirms the creation of the pipeline branch. A quick animated transition adds interface decorators to the components of the newly created pipeline and completes the fade-in from transparent to opaque. All CCP operators of the newly created view and its pipeline link are initialized to default values. For instance, the newly created branch in Figure 4.6(d) does not include the labels that have been added in the existing view, nor have any items been filtered from the view. This is because these labels and the filtering operation are not included in the default presentation state. The pipeline branch creation operation is cancelled by relinquishing the touch anywhere within the bounds of the CCP icon. While the touch point is inside the CCP icon, the new pipeline is

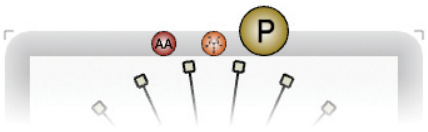


Figure 4.7: The CCP icons bordering the top of a view-pane. Here the presentation CCP icon has been selected, scoping all interactions to the presentation state.

completely transparent, and therefore cancellation occurs in a seamless and non-disruptive fashion.

4.3 Lark Interactions

Lark's collaborative environment contains a integrated meta-visualization and a data visualization, and both are interactive elements. Interactions with the meta-visualization exclusively handle the creation of new views which contain data visualizations. Through use of the meta-visualization one can choose exactly where in the pipeline a particular view should be linked. Interactions with the data visualization are performed directly on the view, allowing collaborators to explore the visualized data set.

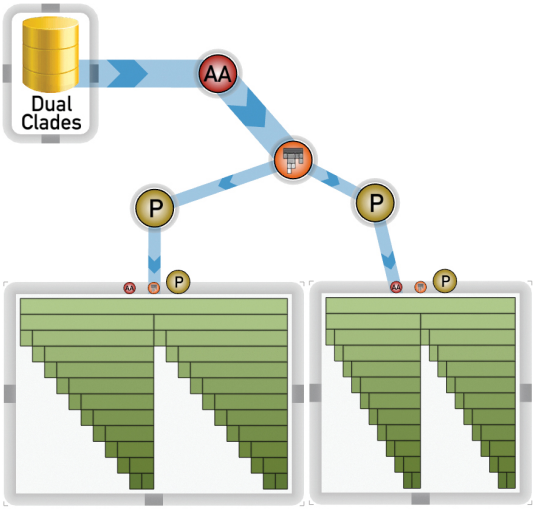
4.3.1 Setting Interaction Scope

All interactions with the visualized hierarchical data happen directly in the view-pane. The interaction scope is explicitly set through three CCP icons bordering the top of a view-pane, as illustrated in Figure 4.7. Note that these bordering icons use the same representation as the CCP icons from the pipeline meta-visualization which connect the data set to the view-pane. The icon for the currently selected interaction scope is displayed at a larger size. For instance, in Figure 4.7 the presentation icon is currently selected indicating that all interactions will target the presentation transformation of the pipeline (see Figure 4.4(a)), altering the presentation state. View transformations are always independent; this means that operations such as changing rotation, translation, and size of a view do not affect other views (as discussed in Section 3.3.9). Hence, the view-pane does not include an icon for view transformations as these transformations are implicitly localized.

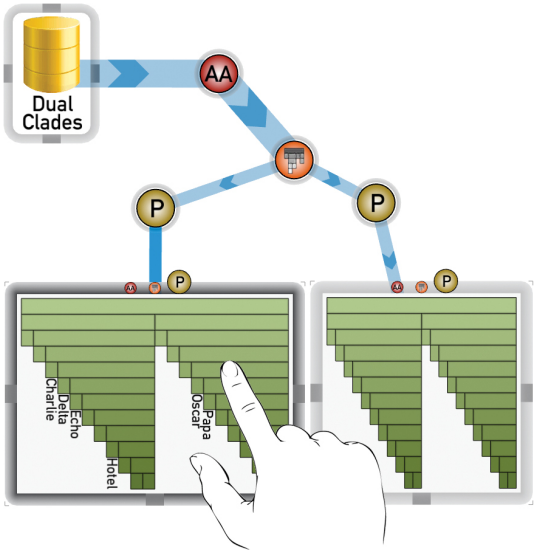
The interaction sequence in Figure 4.8 illustrates the user experience of interaction scoping. Figure 4.8(a) presents two view-panes connected to one another at the spatial layout CCP. The interaction scope for both views are set to the presentation state, as indicated by the larger bordering "P"

icon. In Figure 4.8(b) the analyst has added labels to the view on the left and is about to modify the colour scale. The pipeline edge connecting the presentation CCP to the view-pane is highlighted communicating the scope of the interaction the analyst is about to perform. Since each view-pane has distinct presentation states, these changes are localized to the left view-pane that is being directly interacted with. In Figure 4.8(c) the analyst changes the interaction scope of the right view-pane to the spatial layout by pressing the bordering spatial layout icon. The two view-panes share the same visualization pipeline up to and including the spatial layout, therefore, any modification to the spatial layout pipeline state will affect both views. Again, notice the highlighted pipeline edge communicating the new interaction scope. Through a flick gesture on the right view-pane, the analyst changes the tree layout, as shown in Figure 4.8(d). This interaction affects both view-panes since the operation is modifying the common spatial layout pipeline state. This example illustrates how interaction scoping is controlled through the view-pane CCP icons. Group members can explicitly coordinate how their interactions affect other views in the workspace, realizing the design challenge of *scoped interaction* (see Section 3.3.7).

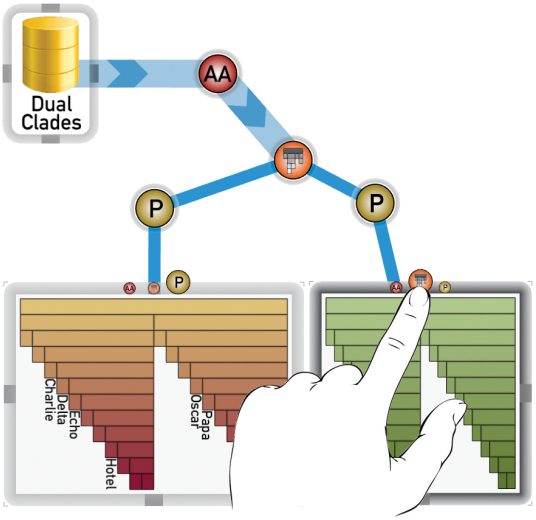
Different visualization pipeline states are modified by setting the interaction scope to target that specific region of the pipeline. The selected interaction scope (i. e., which pipeline state an operation will affect) dictates how interactions will affect the final view. Similar to work by Chi and Riedl [1998], as introduced in Section 2.4.1.3, the distinction between view and value operations is particularly crucial in this context. For instance, a distinction is made according to where filtering occurs in the pipeline: filtering as a data transformation at the analytical abstraction state removes the filtered parts of the data before it reaches the layout, what Chi and Riedl call *value-filtering*, whereas filtering at the presentation state is *view-filtering*. Value filtering occurs before the spatial mapping stage, and hence, will influence the way the data is laid out in space. View filtering will still remove the selected aspect of the data but the basic spatial mapping is left unaffected, with a gap where the data has been removed. This distinction was discussed previously in Section 2.4.1.3. Figure 4.9 illustrates this concept once again, this time within Lark's visualization environment. Here, the same filtering operation is applied to two different points in the visualization pipeline. While filtering in both cases uses the same gestures and has the same general result of removing data, where it is applied in the



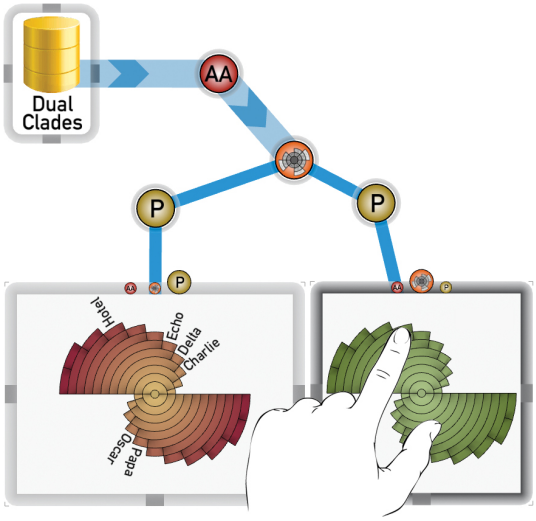
(a) Two view-panes connected to one another at the spatial layout CCP.



(b) The analyst has added labels to the left view and is about to modify the colour scale. The pipeline edge connecting the presentation CCP to the view-pane is highlighted communicating the scope of the interaction the analysts is about to perform.

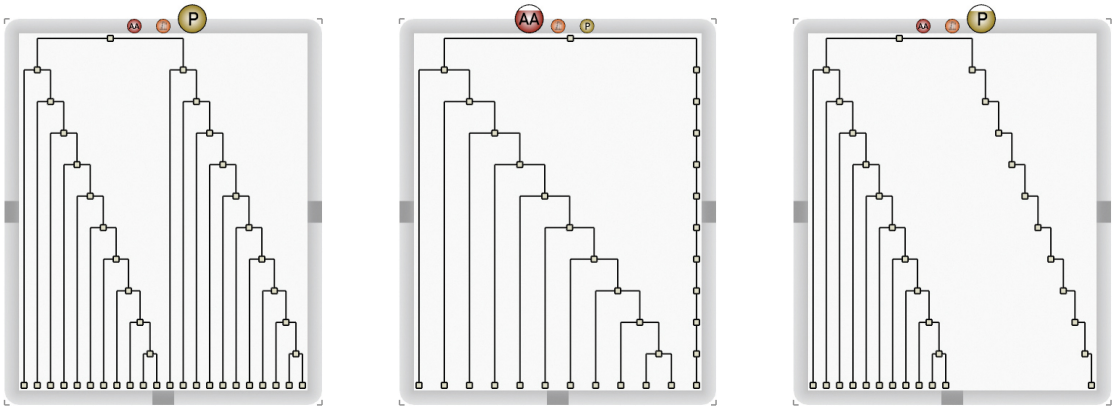


(c) The interaction scope of the right view-pane is changed to the spatial layout. The pipeline is highlighted indicating the extent of the new interaction scope.



(d) The tree layout is changed via a flick gesture directly on the view-pane, altering the common spatial layout pipeline step and therefore changing the data representation in both views.

Figure 4.8: An example interaction sequence illustrating the user experience of scoped interaction.



(a) Complete data set, no items are being filtered.

(b) Filtering at the analytical abstraction CCP (*value-filtering*) .

(c) Filtering at the presentation CCP (*view-filtering*).

Figure 4.9: The outcome of filtering operations applied to different points in Lark's pipeline. The meta-visualization has been omitted for clarity.

pipeline matters, leading to dramatically different results.

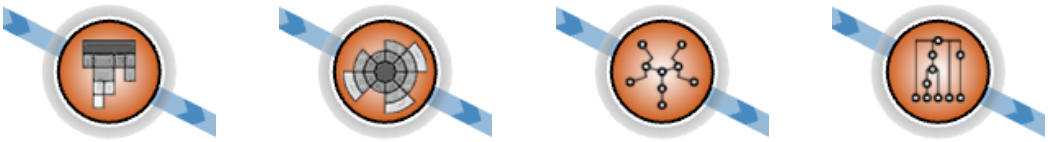
In the development of Lark, an alternative to explicitly setting the interaction scope was explored. For instance, an expanded gesture vocabulary could be developed that would provide each operation its own unique gesture, avoiding the reuse of gestures that are contextualized based on the defined interaction scope. However, informal user feedback suggested keeping the gesture set simpler and using the CCP icons on the view-pane to make the scoping—either analytical abstraction, spatial layout, or presentation—explicit. Furthermore, the explicit nature of setting the interaction scope provides workspace awareness to collaborators through consequential communication and monitoring [Gutwin and Greenberg, 2000].

4.3.2 Coordinated Interactions

To illustrate Lark's coordinated interactions, let us look at a concrete example. Returning to Figure 4.1, we see four view-panes attached to the same data source. Each view shares a common analytical abstraction step, the three icicle plots on the right share the same layout step, and two of these share the same presentation step. By touching the "`P`" icon on the icicle plot at the top right we set this view to receive interactions affecting presentation. Any interaction on this view will automatically affect the other icicle plot that shares the same presentation CCP in the pipeline. Presentation

(a) 100%. (b) 65%. (c) 35%. (d) 23%.

Figure 4.10: Icon meta-visualization of the analytical abstraction CCP icon. The amount of colour fill indicates the percentage of unfiltered items at this state. This same encoding is used in the presentation CCP icon.

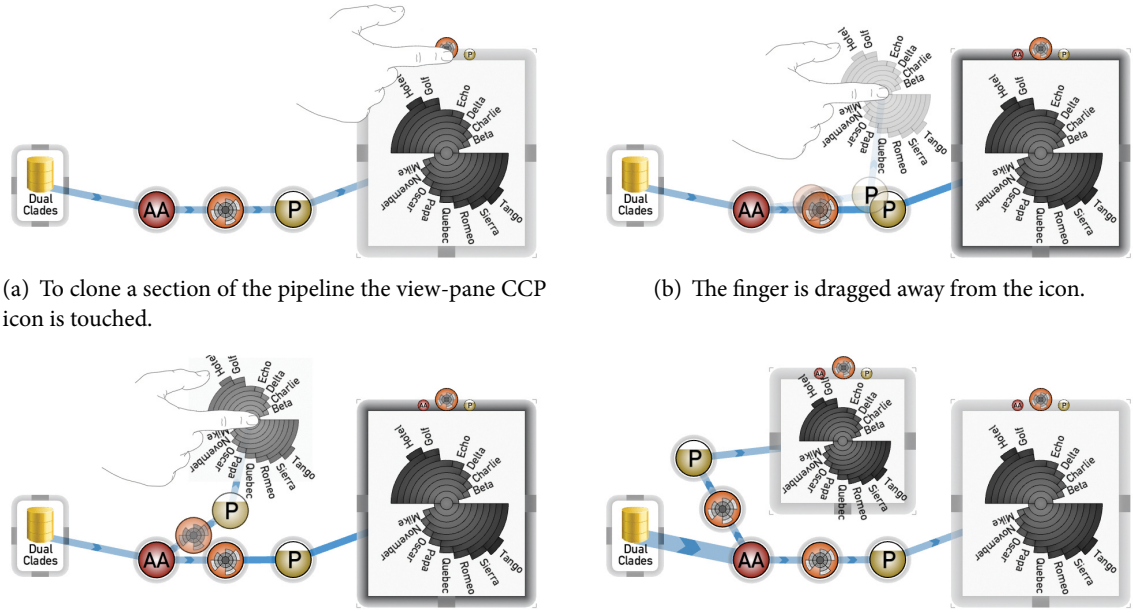


(a) Icicle tree layout. (b) Sunburst tree layout. (c) Radial cladogram tree layout. (d) Cladogram tree layout.

Figure 4.11: Icon meta-visualization of the spatial layout CCP showing the four types of spatial layouts available in Lark.

operations include filtering, colourization, vertex/edge annotations, and selective labelling. Here we can see that both icicle plots connected at "P" share the same colour and labels and have been view-filtered to show the same subset of vertices. The three icicle plots share the same spatial layout CCP. When the spatial layout icon on the view-pane of one of these three views is selected, a gesture can be used to switch the layout of all three views. Four tree layouts are available in Lark: cladogram, radial cladogram, radial-space filling, and icicle plot, as shown in Figure 4.3. The radial cladogram is only connected to the three other views at "AA" and hence has independent layout and presentation parameters. Should the "AA" icon be selected on any of the four views (setting the interaction scope to the analytical abstraction state), a performed filtering operation would affect all four views, changing the spatial arrangement of the unfiltered subset.

Lark includes several awareness features to help establish the coordination of interactions more explicitly. When a view-pane icon is touched and an interaction scope is selected, the pipeline edge connecting this view-pane with others that will be affected, is highlighted. This feature was illustrated previously in Figure 4.8. Similarly, at the start of an interaction gesture, the edge highlights to indicate



(a) To clone a section of the pipeline the view-pane CCP icon is touched.

(b) The finger is dragged away from the icon.

(c) As the distance increases, the cloned pipeline branch gains opacity.

(d) On release, the creation of the cloned branch is confirmed and interface decorators are added to the view-pane.

Figure 4.12: Interaction technique for cloning an existing pipeline branch.

that an interaction is about to occur and which other views it will affect. This helps to maintain common ground among collaborators by explicitly indicating the shared objects in the workspace, as also discussed by Chuah and Roth [2003]. In addition, the pipeline CCP icons are indicative of previous operations on the views connecting them. Both "P" and "AA" icons show the percentage of data that has been filtered (see Figure 4.10), while the layout icon shows which tree layout is currently selected (see Figure 4.11). This gives a high-level overview of what's happening at that particular CCP. Since the encoded metadata is intended to provide a quick overview of a state, encoding has been restricted to a single variable that is most important for that state.

4.4 Pipeline Cloning

The concept of pipeline cloning arose from observation of two different information task scenarios. One occurs during individual work and one occurs between collaborators. The individual scenario involves reaching a point in a data analysis sequence where one would like to compare a variation of one's current view to the current view. The collaboration scenario involves a team member being

interested in a particular view and deciding to commence work starting from that view. Both of these scenarios lead to the need for an independent—not linked—but identical view. Creating new pipeline branches off a CCP icon is sometimes insufficient, as newly created CCPs are initialized to an unaltered default state. The cloning feature addresses this need to duplicate an existing pipeline branch. Cloning a section of the pipeline makes a deep copy of each of the pipeline states, beginning with the selected state and moving down the pipeline to include the view state. This allows one to explore alternative configurations of the pipeline, based on previous modifications. Note that a new branch differs from a clone in that each of the pipeline states of a new branch are initialized to some default value, while a clone is a duplication of the selected pipeline.

A similar interaction technique to the one used in creating a new branch is employed in cloning. Instead of interacting with the CCP icon on the pipeline visualization (as done when creating a *new* pipeline branch, see Section 4.2.2), clones are created by interacting with the CCP icons bordering the view-pane (see Figure 4.7) . Pressing on a view-pane CCP icon and employing the same *touch-drag-release* to outside of the view-pane boundary creates a new clone of a section of the pipeline. This interaction sequence is illustrated in Figure 4.12. The section of the pipeline to be cloned is chosen from whichever CCP icon it is operated from. As the touch is dragged further away from the view window, the opacity of the cloned branch proportionally increases. Confirmation of the operation is made by releasing the touch anywhere outside of the view-pane's boundary. Cancellation of the operation is done by releasing anywhere inside the view-pane. Notice in Figure 4.12(d) that the cloned pipeline branch also includes the filtering and labelling operations of its parent.

Informal user feedback suggested that when initiating a cloning operation, people expected that the same CCP as the selected view-pane icon would be the branching point of the cloned branch. Currently, this CCP is the first pipeline state to be cloned and the preceding CCP is the branching point of the newly created pipeline. For example, in Figure 4.12, the spatial layout CCP icon is touched to initiate the cloning operation (see Figure 4.12(a)). This results in the creation of a cloned pipeline, branching from the "AA" CCP, and the spatial layout state is the first cloned CCP in the newly created branch (see Figure 4.12(d)). The interaction that was expected is that initiating the cloning operation from the spatial layout CCP icon would result in a cloned pipeline branching from

the spatial layout state, and the first cloned CCP is the presentation state. Informal comments from people who tried Lark suggested that this would be the more intuitive result of the cloning operation. Adjusting Lark's implementation to follow this more intuitive outcome would require the addition of a data set icon to the bordering CCP icons around the view-pane. Without this addition there would be no way to clone the complete pipeline branch. Also, this modification would allow the view-pane to be cloned, a feature that is currently not supported by Lark's interface.

4.5 Summary

In this chapter I introduced Lark's interface and interaction design as seen from the perspective of someone using the system. This discussion began with an overview of Lark's visualization environment (Section 4.1), deconstructed into the visual design of the view-panes and the meta-visualization, Section 4.1.1 and Section 4.1.2 respectively. Lark's meta-visualization is structured after the underlying visualization pipeline; a juxtaposed conceptual pipeline and the pipeline represented via the meta-visualization are presented in Figure 4.4. Section 4.2 moves into describing Lark's interaction design. New views and branches off existing pipelines are created with a *touch-drag-release* interaction technique. Since Lark was developed for use on a large tabletop display, this interaction technique is particularly well suited for a touch-sensitive display. Section 4.3 overviews the available interactions with the hierarchical data visualization that are contained within view-panes. This overview is connected with the concept of *scoped interaction* and Section 4.3.1 describes how this is achieved within Lark. Next, the coordination of interactions within the visualization workspace is presented (Section 4.3.2), along with further details of the visual representation of the meta-visualization. Section 4.4 introduces *pipeline cloning*, the last interaction design concept. Here, the motivation for the ability to duplicate specific sections of an existing pipeline is presented and the interaction technique for doing so is described and illustrated.

Building from this overview of Lark's interface design, the next chapter investigates the underlying software architecture design. This succeeding chapter describes the internal software mechanics and engineering principles that made Lark's implementation possible.

Chapter 5

Implementation of Lark

In the previous chapter, I discussed Lark's visual and interaction design, as seen from the perspective of a person who is using the system. To use a automobile analogy, in this chapter I will *look under the hood*, presenting Lark from a system architecture and development perspective. Our discussion will also include the technical challenges encountered and the decisions made during Lark's implementation.

As mentioned in Section 2.2, the utility of the visualization reference model ranges from conceptual to pragmatic. Early work by Haeberli [1988] and Upson et al. [1989] used the data flow model both as a means to organize their systems external visual interface, as well as the system's internal software architecture. We continue with this tradition in architecting Lark: the visualization pipeline is the structural model within the visual workspace as well as throughout the design of the underlying system architecture. This implementation approach realizes Lark's software engineering design challenge of creating a *visualization pipeline-centric software architecture*, as discussed in Section 3.3.10. To support this pipeline-centric design, I design and implemented two libraries—Elm and SnowMonkey—to serve as integral components in Lark's system architecture. Lark, as an Information Visualization (InfoVis) application, integrates the Elm and SnowMonkey libraries with existing technologies from the Large Display Framework (LDF) [Isenberg et al., 2006] and the OpenGL graphics library [Shreiner and The Khronos OpenGL ARB Working Group, 2009]. The implementation of these three software components—Elm, SnowMonkey, and Lark—total over 86,000 lines of code.

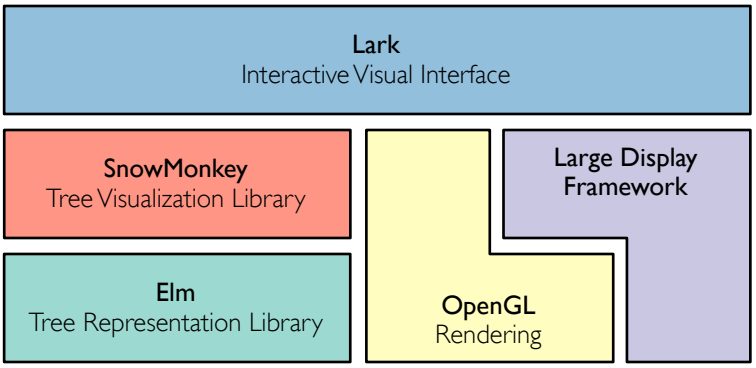


Figure 5.1: Lark's system architecture stack. The Elm, SnowMonkey, and Lark components where developed specifically in order to realize the larger Lark system, in addition to utilizing existing technologies from the Large Display Framework and OpenGL.

Figure 5.1 illustrates how these components are interconnected to form Lark's system architecture stack.

Each of the components in the architecture stack will be discussed in detail within this chapter, with the exception of OpenGL and LDF. OpenGL is a well established and extensively documented [Shreiner and The Khronos OpenGL ARB Working Group, 2009; Wright et al., 2007] computer graphics library whose discussion is beyond the scope of this thesis. Although LDF has been described in previous work [Isenberg et al., 2006], it is not as well known when compared to a library like OpenGL; therefore, LDF will receive a cursory overview.

This chapter is structured as follows. We begin with a high level discussion of Lark's system architecture and its close relation to the visualization pipeline. We then use the structure of Figure 5.1 to organize the presentation of the components in the system architecture, following a bottom-up approach. The design of the Elm library is introduced and its implementation details are discussed. Next, the SnowMonkey library is presented, along with a discussion focusing around the separation of pipeline states within the application programming interface (API). Lastly, the Lark application, found at the top of the architecture stack, is discussed. The chapter is concluded with an overview of Lark's implementation details.

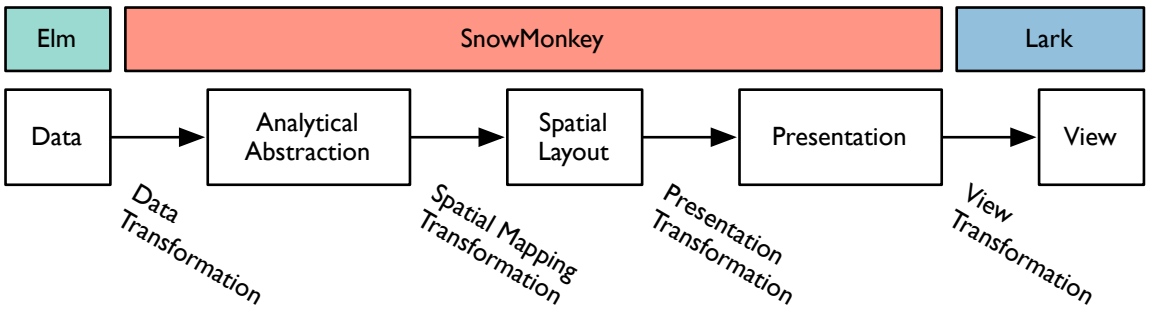


Figure 5.2: The visualization pipeline and the specific software components which implement sections of the pipeline. The same colourization of software components from Figure 5.1 is used in this diagram.

5.1 Lark's System Architecture and The Visualization Pipeline

The software architecture design of Lark closely resembles the visualization pipeline. This theoretical pipeline models the transformation of raw data into interactive computer graphics as a series of chained data processing steps. Architecting software after the visualization pipeline means that data processing occurs sequentially through a series of discrete transformations, and the movement of processed data from one pipeline step to the next is well defined and separated in the API.

Figure 5.2 illustrates the specific software components from Lark's system architecture stack which implement the various sections of the visualization pipeline. At the beginning of the pipeline implementation is the Elm library, which is solely responsible for providing a generic hierarchical data structure. Elm implements the *data* state of the visualization pipeline. The SnowMonkey library transforms the hierarchical data from Elm into a visual presentation. SnowMonkey implements a section of the visualization pipeline from the *data transformation* through to the *presentation* state. Each of the pipeline states in SnowMonkey are implemented as discrete, data caching modules. The final component in the pipeline implementation is the Lark application which covers the *view transformation* and the *view* state. Lark uses OpenGL for rendering the view and LDF for the user interface.

The naming convention of Lark's software components is based on flora and faunae: a tree, a primate, and a bird. At the bottom of the architecture stack is a tree library, appropriately named *elm*, the common name of the genus of deciduous and semi-deciduous trees commonly found in the

northern hemisphere [Encyclopedia of Life, 2010c]. Next, the tree visualization library gets its name from the Japanese macaque, or *snow monkey*, the northernmost living primate apart from Homo sapiens and an Old World monkey who makes its habitat in the highlands and forested mountains of Japan [Encyclopedia of Life, 2010b]. Lastly, *larks* are a medium-sized bird from the order Passeriformes (commonly known as "perching birds" [Edwards and Harshman, 2008]) and are primarily found in the Old World—Europe, Asia, and Africa [Encyclopedia of Life, 2010a]. The idea behind these creatures is that Lark's software begins with trees, expands to primates that live in forests, and moves on to birds that live in forests and beyond.

5.2 Elm: Tree Representation Library

At the bottom of the architecture stack is Elm, a software library for representing *hierarchically structured data*, often simply referred to as *trees* (as introduced in Section 2.5). It handles the management of the hierarchical structure, while providing flexibility as to the composition of the hierarchy. Elm is a generic, standalone library which implements the initial *data* state of the visualization pipeline, as shown in Figure 5.2. It provides a structure to the raw data which can be accessed from subsequent states in the pipeline. The structure and access mechanisms are optimized for performance in order to avoid potential performance bottlenecks in the subsequent states.

The development of Elm was necessary in order to satisfy the need for an uncompromising and consistent manifestation of the pipeline-centric design throughout the different tiers of the Lark system's API, beginning with the lowest tier. This is why existing software libraries for data hierarchies, such as the Boost Graph Library [Siek et al., 2002] and the `tree.hh` Library [Peeters, 2010], were not used.

Elm is written in C++ and supports generic data types through the use of templates and multiple inheritance. The core Elm library contains 6,408 lines of code and the accompanying unit tests contain 3,560 lines of code. The individual C++ classes which make up the core Elm library are illustrated in Figure 5.3, using UML class diagram notation. Note, that while the relationships between classes are accurately reflected in the diagram, the full API specification for each of these classes has been simplified (e. g., only 16 of the 31 defined methods in the `Tree` class are shown). The smallest

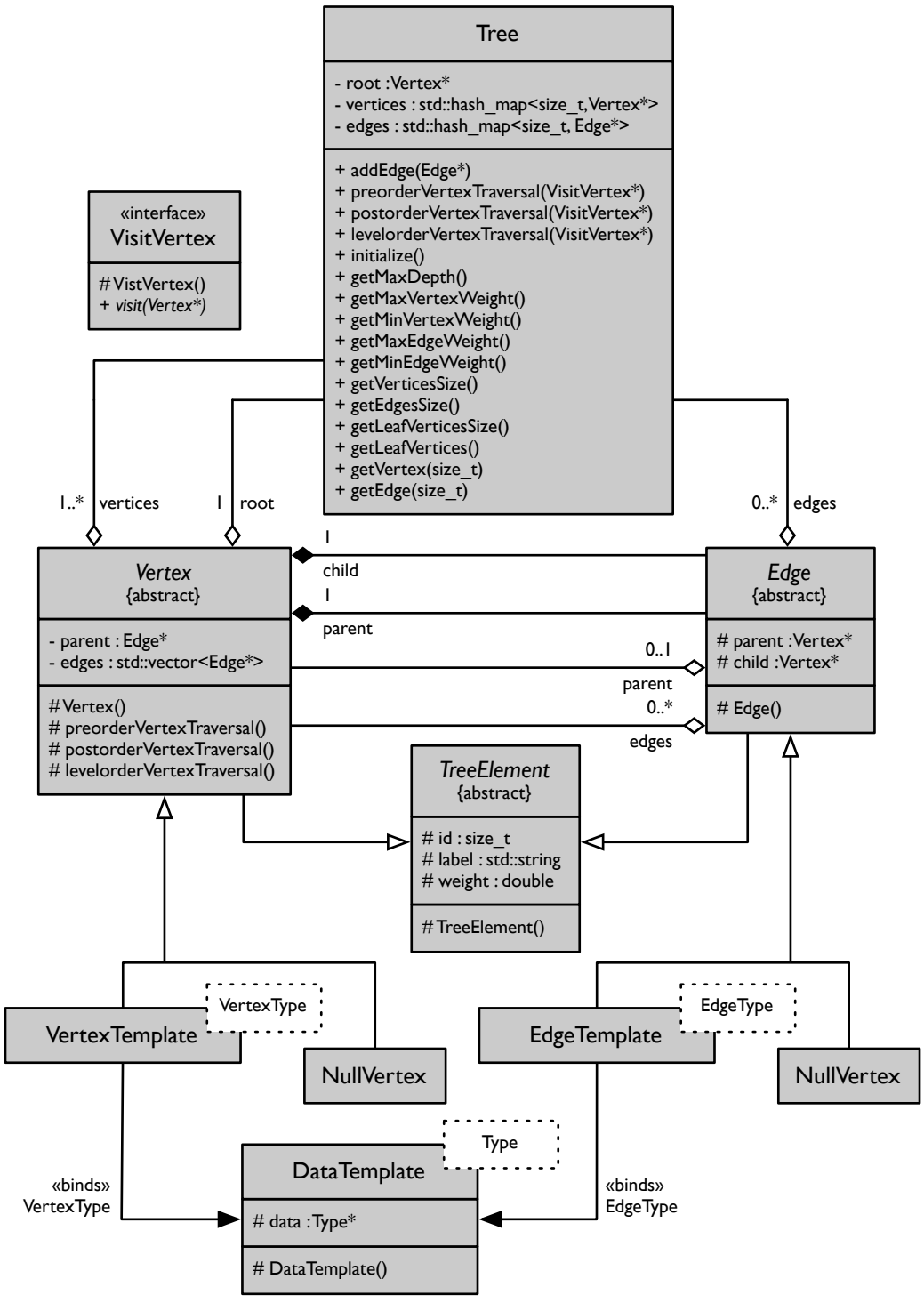


Figure 5.3: An overview of the individual classes which comprise the Elm library and their relationship to one another, illustrated using UML class diagram notation.

components of Elm are the `Edge` and `Vertex` classes. An `Edge` links two `Vertex` objects together and a `Vertex` stores an adjacency list of child `edges`, as well as its connecting `parent` edge. This bidirectional tree implementation enables navigation up and down the hierarchy beginning from either an `Edge` or a `Vertex`. `Tree` objects control the management of all the individual `Edge` and `Vertex` objects which comprise to form the definition of a single hierarchy. The `Tree` class is the entry point for performing preorder (depth-first), postorder, and level-order (breadth-first) tree traversals, as well as querying other tree attributes, such as the maximum depth of the tree, *etc.*. The `vertices` and `edges` hash maps contained in `Tree` are used for quick lookups of vertices and edges based on a unique `id` identifier, an attribute of the parent `TreeElement` class of `Vertex` and `Edge`. `TreeElement` also provides `label` and `weight` attributes. The latter attribute provides support for weighted trees and can be used in defining an evolutionary tree, also known as a *phylogenetic tree*, where edge weight stores the time between speciation events [Baum and Offner, 2008].

All the container classes (e. g., `std::hash_map` and `std::vector`) used to store vertices and edges within the Elm library use either the abstract `Vertex` or `Edge` class as the type of the container's elements. The `VertexTemplate` and `EdgeTemplate` classes, derived from these abstract classes, can reference arbitrary objects with the template `data` member, defined within `DataTemplate`. `Tree` objects can therefore contain heterogeneous collections of data types attached via the `VertexTemplate` and `EdgeTemplate` objects that comprise the hierarchy. Listing 5.1 presents an example which illustrates this concept. Notice that the types of tree vertices in this example are a mixture of `NullVertex`, `VertexTemplate<Circle>`, and `VertexTemplate<Square>` data types, afforded by Elm's flexible API. Once all the vertices and edges have been added to the tree, the `initialize()` method is called which computes the meta-data tree attributes, such as the maximum depth of the tree, maximum/minimum vertex/edge weight, *etc.*. Extracting `data` object references from the tree requires type casting via a `dynamic_cast`.

Tree traversal in Elm is performed via the `VisitVertex` interface. Objects that implement this interface are passed in as arguments to one of the vertex traversal methods—preorder, postorder, and level-order—in `Tree`. The traversal methods navigate the tree, visiting each of the tree vertices in the appropriate order, calling the `vist(Vertex*)` method on the passed in `VisitVertex` object for

each traversed `Vertex` object. The implemented `visit(Vertex*)` method performs the necessary processing on the `Vertex`, accessing the parent `Edge` if necessary.

```
class Square {
    double length;
public:
    Square(double l) : length(l) { }
};

class Circle {
    double radius;
public:
    Circle(double r) : radius(r) { }
};

int main() {
    Elm::NullVertex* root = new Elm::NullVertex();
    Elm::Tree* tree = new Elm::Tree(root);
    /* Create circle vertices and square edges. */
    for(int i = 0; i < 4; i++){
        Elm::VertexTemplate<Circle>* vertex =
            new Elm::VertexTemplate<Circle>(new Circle(5.0));
        Elm::EdgeTemplate<Square>* edge =
            new Elm::EdgeTemplate<Square>(root, vertex, new Square(4.0));
        tree->addEdge(edge);
    }
    /* Create square vertices and circle edges. */
    for(int i = 0; i < 4; i++){
        Elm::VertexTemplate<Square>* vertex =
            new Elm::VertexTemplate<Square>(new Square(5.0));
        Elm::EdgeTemplate<Circle>* edge =
            new Elm::EdgeTemplate<Circle>(root, vertex, new Circle(8.0));
        tree->addEdge(edge);
    }
    tree->initialize();
    return 1;
}
```

Listing 5.1: Example C++ source code illustrating how a tree of heterogeneous data is created using the Elm library. Here a tree with eight leaf vertices—four `VertexTemplate<Circle>` and four `VertexTemplate<Square>` objects—at a depth of one is created and initialized.

Since the initial release of the C++ version of Elm in early 2008, the library has been ported to both Java and C#, released under the names Elm4J and Elm4Cs respectively. Elm4J was publicly re-

leased under the GNU Lesser Public License [Free Software Foundation, 1997] on May 4, 2009 and is available for download from http://innovis.cpsc.ucalgary.ca/Software/Elm4J. Both Elm4J and Elm4Cs include a New Hampshire tree format, or simply *Newick format* [Wikipedia, 2010b] file parser which is found outside of the core Elm library in the C++ version.

5.3 SnowMonkey: Tree Visualization Library

The SnowMonkey library builds on Elm, implementing the *data transformation* to *presentation* state section of the visualization pipeline (see Figure 5.2). The goal of SnowMonkey's pipeline implementation is to keep different pipeline states separate and discrete. This is achieved via two main features: *caching* encapsulated data from previous pipeline states, and the addition of *pipeline state specific meta-data* at each pipeline transformation. Data is processed from one pipeline state to the next by encapsulating and making a cache of this incoming data, applying the pipeline transformation to the cache, and then adding any pipeline transformation specific meta-data. For example, when data items (comprised of vertices and edges) move from the *analytical abstraction* state to the *spatial layout* state, the *spatial mapping transformation* augments these items with geometrical meta-data, assigning each vertex and edge a position, size, and shape. Caching makes it possible for later pipeline states to modify definitions made by earlier states. The pipeline transform operates on the cache rather than the encapsulated data, meaning the data defined at an earlier state is left unaltered. For example, the presentation transformation handles emphasizing specific data items. This emphasis might involve modifying the position and size of data items (e. g., fish eye distortion), two properties that were defined previously by the spatial mapping transformation. Since this emphasis operation within the presentation transformation modifies a cache of the data items position and size, the underlying value defined by the spatial layout transformation is left unaltered. Furthermore, this is an important feature as the visualization pipeline must support an arbitrary number of branches at any pipeline state. Since pipeline transformations never modify the data defined at previous pipeline transformations, multiple pipeline branches do not interfere with each other.

SnowMonkey is written in C++, building on the Elm library which is its only dependency. The core SnowMonkey library contains 30,310 lines of code and the accompanying unit tests contain

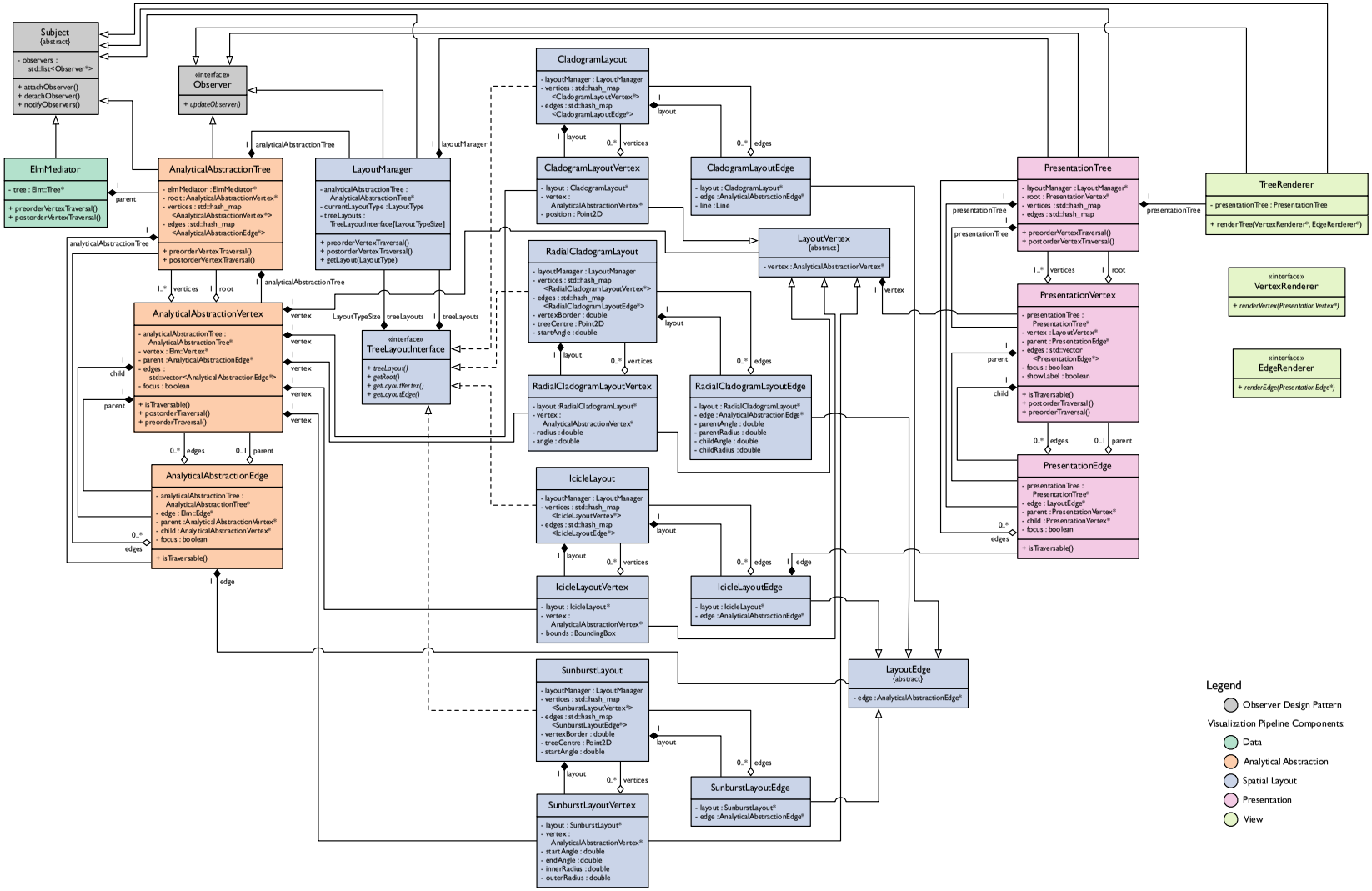


Figure 5.4: An overview of the individual classes which comprise the SnowMonkey library and their relationship to one another, illustrated using UML class diagram notation.

16,911 lines of code. The individual classes which make up SnowMonkey are illustrated in Figure 5.4, using UML class diagram notation. This diagram has been annotated with the classification of individual classes according to the specific stages of the visualization pipeline that they implement. Coordination of changes occurring at each pipeline state is done through use of the observer design pattern [Gamma et al., 1995, p. 293], which notifies subsequent pipeline stages that the encapsulated data has been modified and the caches must be updated.

Notice in Figure 5.4 that the SnowMonkey library includes classes labelled as data and view state components, contrary to the range of implemented pipeline components as indicated in Figure 5.2. These components are wrappers (e. g., the adapter software design pattern [Gamma et al., 1995, p. 139]) between the different library APIs. For example, the `SnowMonkey::ElmMediator` (located on the far left side of Figure 5.4) wraps an `Elm::Tree` object, inheriting from the `SnowMonkey::Subject` class such that the observer design pattern can be used with `Elm::Tree` objects within SnowMonkey. This design makes use of the mediator software design pattern [Gamma et al., 1995, p. 273]. Similarly for the `SnowMonkey::TreeRenderer` object, seen on the far right side of Figure 5.4. Moreover, these data and view state components do not implement any of the pipeline state data processing logic, but rather function as necessary interfaces between the different APIs.

As mentioned previously, the goal of SnowMonkey's pipeline implementation is to keep the different states separate and discrete from one another. Figure 5.4 illustrates this division between pipeline states. Each state references objects of the preceding pipeline state and succeeding pipeline states are never referenced. For example, `SnowMonkey::AnalyticalAbstractionTree` contains a reference to `SnowMonkey::ElmMediator`, which in turn references `Elm::Tree`, however `SnowMonkey::ElmMediator` has no reference to `SnowMonkey::AnalyticalAbstractionTree`. This discrete layering of pipeline states allows for the creation of numerous pipeline branches which build upon preceding pipeline states. Furthermore, separation between pipeline states (e. g., such that changes to the presentation state does not affect the spatial layout state) is integral in realizing algorithmic support for *scoped interaction* throughout the pipeline. Changes to a particular pipeline state are localized to that state, affecting the subsequent but never preceding pipeline states.

5.4 Lark: Interactive Visual Interface

At the top of the software architecture stack is Lark, the application which ties in SnowMonkey with LDF and OpenGL to create the visualization workspace, as seen throughout Chapter 4 (e. g., Figure 4.1). LDF is an interaction framework which uses an underlying buffer concept [Isenberg et al., 2006] to enable scalable, interactive response of interface components for large displays. The framework handles object management via a scene graph, and like Lark, uses OpenGL for rendering. Lark is also responsible for integrating the end products of SnowMonkey's visualization pipeline into LDF's API for visual interfaces and rendering this geometry with OpenGL. Lark leverages LDF to provide *spatial flexibility* for all visual components within the workspace, allowing components to be freely oriented and positioned throughout the workspace; as LDF provides the implementation of interaction techniques such as integrated rotation and translation (RNT) [Kruger et al., 2005]. Lark is decoupled from SnowMonkey such that it would be straightforward to use different windowing toolkits to implement the end user interface of the system, while keeping all the back end functionality provided by Elm and SnowMonkey. This ability has already been exercised, as Lark is available under Microsoft Windows using the Qt user interface framework [Nokia Corporation, 2010] and Mac OS X using the Cocoa framework [Apple Inc., 2010]. The Lark application measures contains 29,591 lines of code.

5.5 Summary

In this chapter I introduced the underlying software architecture of the Lark system. I discussed how Lark was architected after the visualization pipeline, with a well defined separation between the different pipeline states within the API. The remainder of this chapter was organized after Figure 5.1, which illustrates the software components which make up Lark's system architecture stack. In Section 5.2, I introduced the Elm library and discussed its implementation details. Elm is a generic library for hierarchical structured data, originally written in C++ and later ported to both Java and C#. I then introduced SnowMonkey in Section 5.3, presenting an overview of the library. Next, the Lark application is discussed in Section 5.4, which integrates the SnowMonkey library with LDF and OpenGL into the end user application.

Chapter 6

Conclusion

Real-world information continues to grow in size and complexity, making the analysis and interpretation of this data an ever increasing challenge. Both *information visualization* and *collaborative team work* have been suggested as important factors in addressing these information complexity challenges [Thomas and Cook, 2005]. Information visualization has the potential to provide different ways of examining and exploring the data. Collaborative data analysis can combine the analytic power of multiple individuals, with the possibility of including varying types and levels of expertise, potentially leading to increased quality of solutions and discoveries. However, while considerable research is being conducted in both the areas of Information Visualization (InfoVis) and Computer-Supported Cooperative Work (CSCW), comparatively less research examines the interplay between them. This is especially true for co-located collaborative scenarios. In this work I have strived to bridge this gap, investigating how collaborative data analysis can be supported within an information visualization environment.

In this thesis I have focused on supporting small groups of people working together in a synchronous co-located environment who make use of information visualizations in their analysis process. Within this space I am particularly interested in *mixed-focus collaboration*—team work characterized by frequent changes in collaboration styles, which span the range from loosely coupled, individual work to closely coupled, group work [Gutwin and Greenberg, 1998]. To facilitate mixed-focus collaboration, I investigated how to support changing collaboration styles within a collabora-

tive information visualization workspace. To this end, I identified three concepts—temporal flexibility, spatial flexibility, and scoped interaction—which play an important role in this type of work scenario.

These concepts formed the basis for my research challenges and structured the conceptualization, design, and implementation of Lark: a coordinated multiple views (CMV) visualization environment where the relationships and connections between individual views are illustrated through an integrated meta-visualization [Weaver, 2005]. The meta-visualization is modelled after the visualization pipeline and provides several distinct stages in which group members can coordinate their interactions—these stages are identified as *collaboration coordination points (CCPs)*. Making the relationships and connections between the individual views visually explicit supports workspace awareness. The CCPs can help empower group members with the freedom to work in concert or independently. Furthermore, Lark instantiates a coherent interactive visualization collaboration environment with direct visual and algorithmic support for the coordination of data analysis actions over shared large displays.

In this thesis I present a novel approach to the coordination of interactions between multiple people working together in a co-located collaborative visualization environment. The coordination of interactions can help facilitate mixed-focus collaborative work by supporting both individual and group work, and the transitions between these different levels of collaboration. By investigating the synergy of information visualization and collaborative team work, this thesis demonstrates promising strategies for addressing present day information complexity challenges.

In this final chapter, I summarize the research contributions that I have made in the conceptualization, design, and implementation of Lark. I also discuss future research opportunities leading from this work. Section 6.1 reintroduces the research challenges of this thesis, originally outlined in Chapter 1, and Section 6.2 summarizes the research contributions made in this thesis. To further contextualize my research contributions, Section 6.3 presents an example collaborative data analysis scenario, like the one from Chapter 1, which makes use of Lark in the analysis process. Directions for future research are reflected upon in Section 6.4, and Section 6.5 contains the concluding thoughts of this thesis.

6.1 Research Challenges

Chapter 1 outlined three research challenges when supporting changing collaboration styles for synchronous co-located collaborative work within a shared digital workspace. I will now repeat these challenges and in the following sections describe the extent to which each of these challenges have been addressed in this thesis.

1. **How can we provide temporal flexibility for data analysis activities?** Recent evidence suggests that temporal flexibility among information analysis tasks is common practice among team workers [Isenberg et al., 2008]. I will investigate how to support this behaviour, particularly I will consider supporting concurrent interaction and not requiring any specific temporal flow of activities, thus allowing team members to follow their own unique analysis approaches.

2. **How can we provide spatial flexibility for visualizations and collaborators around the shared digital workspace?** Since changing collaboration cohesion in large workspaces is commonly accompanied by changing team member locations, it is important for artifacts in the workspace to be mobile as well. In my research, I will provide a flexible approach to workspace organization that can allow team members to establish their own work areas [Scott et al., 2004], thus providing support for team members to coordinate their actions [Kruger et al., 2005].

3. **How can we provide scoped interaction?** Empowering team workers with the ability to work in parallel without interfering with each other's task is crucial for collaboration [Scott et al., 2003a]. My research investigates how each interaction within a digital workspace can be scoped in a manner that immediately and persistently informs all workers, thus supporting concurrent and asynchronous interaction.

6.2 Contributions

The contributions that I make in this thesis fall under the area of information visualization for synchronous co-located collaborative work. This work is an attempt to advance the goal of supporting changing collaboration styles within shared digital workspaces. To this end, I have identified three research challenges that this thesis focuses on—temporal flexibility, spatial flexibility, and scoped

interaction—listed above in Section 6.1. My research contributions indirectly and directly address these research challenges. Indirectly, an extensive research effort was invested into designing and implementing a system intended to support synchronous co-located collaboration on large tabletop displays. The results of this investment was the Lark system, which is a contribution in itself. The embodiment of Lark also presents innovative research concepts of its own. Many of these concepts directly address the aforementioned research challenges of this thesis. The following list summarizes both the indirect and direct research contributions that I make in this thesis:

Lark System: The design and implementation of the Lark system, designed to support changing collaboration styles by allowing people to switch between tightly and loosely coupled work within a CMV environment.

Pipeline Meta-visualization: Structuring the coordination between multiple views of a visualization after the theoretical visualization pipeline, and explicitly representing this structure within the collaborative workspace through an integrated meta-visualization.

Collaboration Coordination Points: Using the discrete states within the visualization pipeline as CCPs to explicitly define how views are coordinated with one another.

Workspace Awareness: Using an integrated meta-visualization to provide workspace awareness to collaborators working in a CMV environment. The meta-visualization informs collaborators about the connections and relationships between the individual visualizations.

Temporal Flexibility: Actualization of the concept of temporal flexibility—requiring no specific temporal ordering of activities—within Lark. This allows team members to follow their own unique analysis approaches and leave any number of visualizations open for later investigation. This contribution addresses research challenge No. 1.

Spatial Flexibility: All interface components within Lark environment can be individually placed, scaled, and organized through out the workspace, realizing the concept of spatial flexibility. This flexible approach to workspace usage supports mobility among team members, which is a

common factor in changing collaboration styles [Scott et al., 2003a]. Furthermore, Lark's realization of spatial flexibility allows people to work side-by-side from each other on completely different visualizations, as well as across the table from each other on the same visualization. This contribution addresses research challenge No. 2.

Scoped Interaction: Lark actualizes the concept of scoped interaction by enabling the explicit definition of an interaction's scope via the CCP icons bordering the visualization view-pane. The extent of an interaction's effect is made visually explicit in the integrated meta-visualization, further adding to individual and group awareness. This contribution addresses research challenge No. 3.

History of Visualization Process Models: A detailed literature review of the history of visualization process models is presented in Chapter 2. This survey traced the evolution of the conceptualized visualization process and identified how these ideas have been captured in formalized models.

In addition to the major contributions listed above, there are several minor contributions which came out of the realization of Lark. These interface innovations and software engineering contributions include:

- The use of CCPs to provide light-weight visualizations of the current state of the visualization pipeline to support workspace awareness.
- The ability to dynamically clone sections of the visualization pipeline.
- Explicitly using the visualization pipeline as a branching tree structure, with visualization views at the leaf vertices of the tree.
- Using an integrated meta-visualization to explicitly communicate which views within the workspace are the same.
- The distinction of value and view operations, such as filtering, within Lark's visualization environment.

- The public release of Elm4J, the Java port of the Elm tree representation library, under the GNU Lesser Public License [Free Software Foundation, 1997].

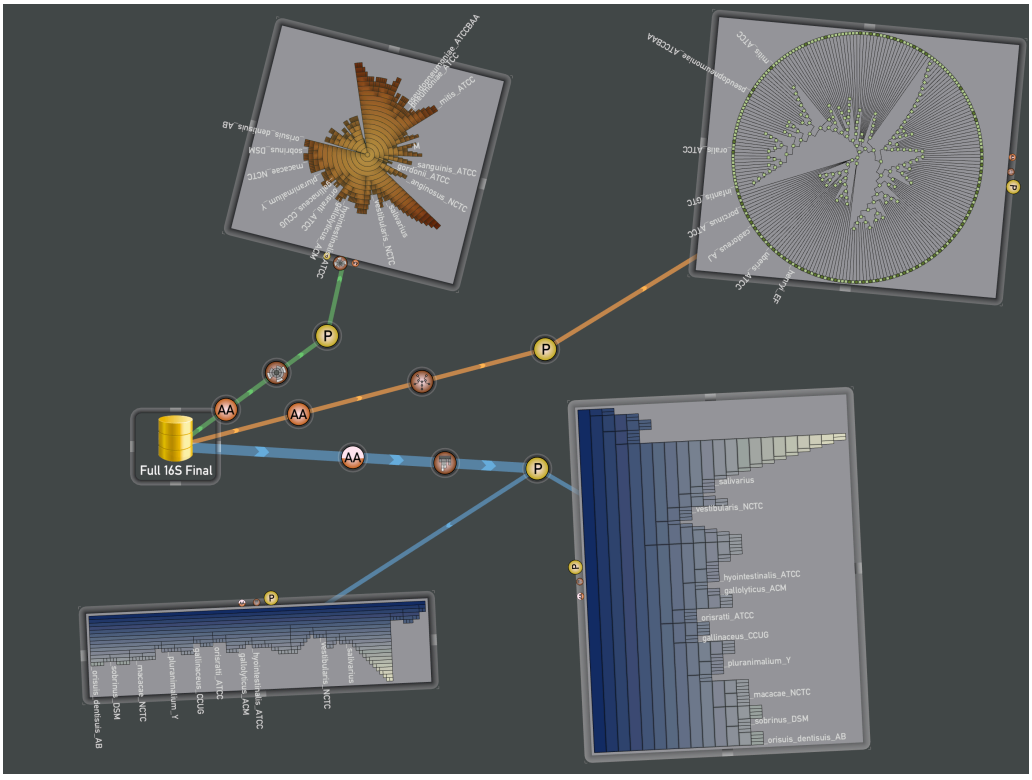
6.3 Example Collaborative Scenario Using Lark

In Chapter 1, to explain the type of collaboration I investigate in this thesis, I described an example scenario which followed a fictitious team of three biologists—Alice, Bob, and Carlos—working together to analyze data from their latest experiment. In this scenario the fictitious team made use of paper-based visualizations and charts of their experiment data, with digital tools being absent from this component of the analysis process. In this section I revisit this example scenario, illustrating how this fictitious team of biologists could conduct their collaborative data analysis in a digital context using Lark. Similarly to the scenario in Section 1.4, the motivation behind the three biologists' collaboration, in a synchronous co-located environment, is due to the large number of genes and the complexity of the biological networks used in this experiment. The following scenario description identifies key aspects of the collaborative process and how these aspects are supported by Lark's collaborative environment.

6.3.1 Parallel Work

The three biologists come together around a large tabletop display, load their data into Lark, and begin their investigation. Since they do not know what to expect from their data, they first enter into an exploratory analysis phase. To broaden their data coverage, they initially decide to explore the data in parallel to look for interesting patterns individually. Alice, Bob, and Carlos all create separate branches from the data source, as shown in Figure 6.1. Alice wants to get an overview of the data and chooses to change presentation details by switching to an appropriate colour scale, adding labels, and filtering out data that she deems uninteresting. Bob takes a different approach and wants to explore the largest hierarchical branch first. He filters on the analytical abstraction state by touching the "AA" icon on the view. By performing a filter gesture he now filters parts of the data, and a new layout is created that only shows the remaining information in a larger size (e. g., value filtering). Carlos creates two different views branched at "Spatial Layout" and decides to do a comparative exploration to see if any representation will help him to see patterns in the data better.

(a) The position of the three biologists around the tabletop display during the parallel work phase.



(b) Screen shot of the visualization workspace during the parallel work phase.

Figure 6.1: Parallel work: Initially Alice, Bob, and Carlos start by exploring the data with entirely separate views of the same data set.

This initial work partitioning outlines some of the collaboration features we want to address. First of all, *spatial flexibility* allows the three biologists to organize themselves around the tabletop display in the way they feel is most appropriate. The area where the biologists conduct their information exploration is of their own choosing, not the system's. On the three sides of the table, the biologists can establish personal territories in which they place their visualization views. By establishing a completely independent *interaction scope* at the data level, all three can work in parallel without affecting each others' views of the data. As to freedom of *collaboration style*, in this scenario, all three team members chose to work in parallel. Lastly, Lark is also designed so that interactions are *temporally flexible*. As illustrated in this scenario, data analysis can be started from many different perspectives: from identifying data items through use of colour and labelling, to concentrating on a subsection of the larger data set by filtering, to exploring different layout options. Also, no global interactions interfere with the sequence each group member chooses.

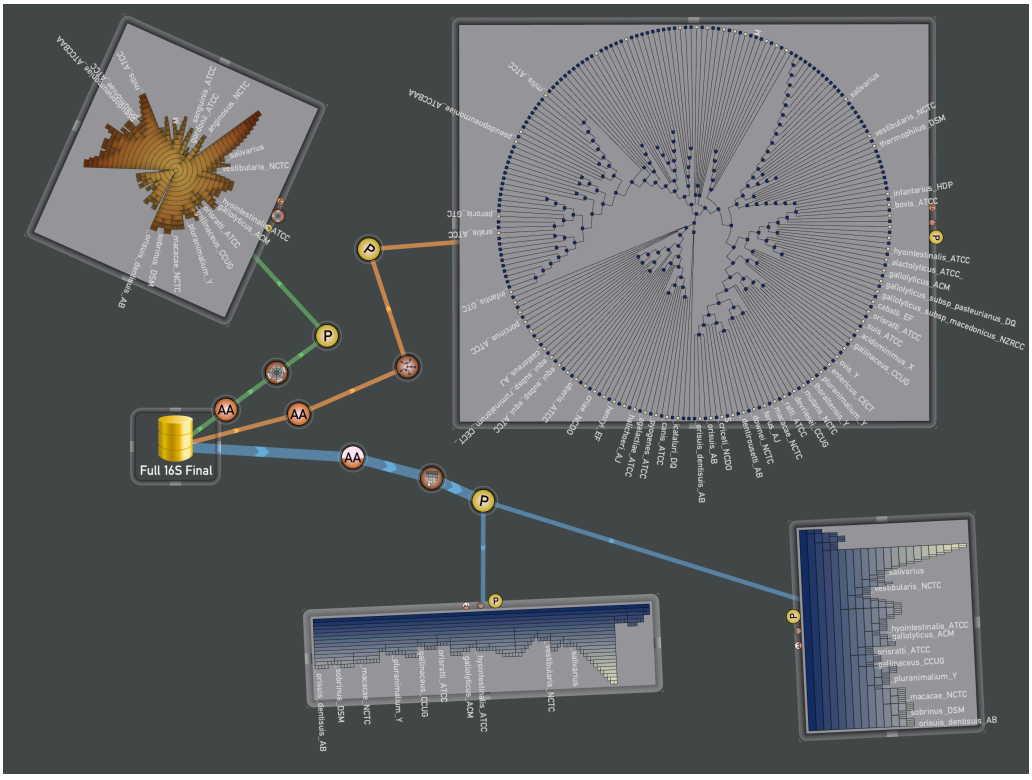
6.3.2 Parallel and Joint Work

While Alice has found some interesting patterns in the data that she wants to examine, Carlos was not as successful with his approach. Carlos walks over to Bob's work area and glances over his shoulder, noticing that Bob is closely examining an interesting branch. To avoid disturbing Bob, Carlos simply creates a new view from "P" and closely watches Bob's interactions. As the work continues, Bob and Carlos start a closer discussion of the data and decide to enlarge one of the views, examining data details together (see Figure 6.2). We see that Alice, Bob, and Carlos are working in different collaboration styles. Alice is in loosely coupled collaboration to the other two, while Bob and Carlos work more closely together. In this scenario, spatial flexibility lets Bob and Carlos reposition and resize the view they are jointly analysing. By specifically establishing a closely linked interaction scope, Carlos can follow Bob's interactions with the data for a while until they join in more closely coupled work. Alice's work remains separate.

6.3.3 Joint Work

As our three team members continue with their work, they create a number of different views of the data that they want to compare and organize in the workspace. When they want to save a certain

(a) The position of the three biologists around the tabletop display during the parallel and joint work phase.



(b) Screen shot of the visualization workspace during the parallel and joint work phase.

Figure 6.2: Parallel and joint work: Bob and Carlos are discussing a view together while Alice still focuses on her own analysis.

state, they clone complete branches and continue their work from the new branch. At some point all three decide to come together to see what they have found, closely discussing and negotiating their findings. Figure 6.3 shows Bob, Carlos, and Alice having moved to one area of the workspace to discuss one view together. As the views that were created in the meantime are still in the workspace, they can see how each team member progressed through the analysis. If the discussion makes closer examination of the data necessary, each view can be further interacted upon.

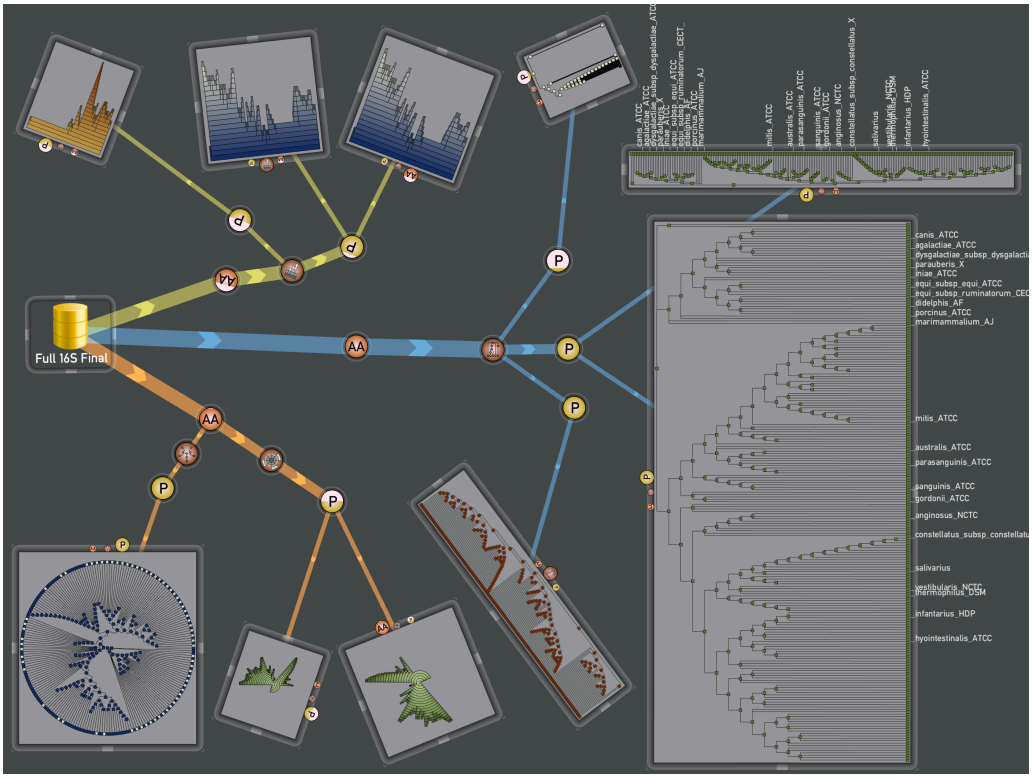
In this example, the biologists returned to previously examined items in discussing their findings. This is an example of temporal flexibility and is supported in Lark, in part by removing the sequencing of actions, and in part by allowing any work paths to be left available within the working environment. Keeping the work paths of this data exploration history, in evidence on the display, provides parallels to the freedoms of working with printed visualizations. The reorganization of both workspace items and biologists around the display is another example of spatial flexibility; here it is used in the transition between different styles of collaboration. In this joint work phase of the collaborative work session, new views and clones of existing views are dynamically created. Scoped interaction is utilized such that the interactions on these views will not unintentionally modify existing views within the workspace.

6.4 Future Work

In this thesis I have strived to advance a research goal that cannot be achieved in a single thesis alone. My thesis is but a few steps towards this goal, and as such, there is much work left to do. The future research opportunities which arise from this thesis are numerous, and in this section I will discuss a few of these potential research directions.

There are a series of improvements and extensions that could be made to Lark's implementation. A commonly requested feature from the people who tried Lark is the ability to merge pipeline branches. Proper support for this system behaviour requires a full suite of set operations for performing merges, such as: union, intersection, and complement. Merging operations could also be generalized to provide algorithmic support for questions such as: "show me all the common elements in these two views." A second commonly requested feature is algorithmic support for automatic spa-

(a) The position of the three biologists around the tabletop display during the joint work phase.



(b) Screen shot of the visualization workspace during joint work.

Figure 6.3: Joint work: Bob, Carlos, and Alice have moved to the same work region and enlarged one view to discuss.

tial organization of the meta-visualization and view-panes. Lark does not organize the workspace in any predefined manner, leaving it up to the people using the system to determine how interface components should be organized. While this system behaviour is important for the emergence of collaborative group dynamics [Tang et al., 2006], an automatic layout feature could also be provided on demand. Another extension would be to support additional data types and operations. In its current implementation, Lark supports the visualization of hierarchical data and provides a compelling, though limited, set of operations on this type of data. This could be extended to support additional types of data and a larger set of available operations on these data types.

It would also be interesting to investigate how Lark's collaboration concepts apply to alternative environments. While Lark was designed for tabletop interaction, it has also been frequently used on a large dual monitor, high-resolution desktop setup. Interestingly, many of the tabletop features such as free rotation for collaboration communication [Kruger et al., 2005], were also used in a similar communicative manner in the desktop setup. This would be an interesting direction for future work. Furthermore, it would also be compelling to explore how Lark could be extrapolated to support distributed or mixed-presence collaborative environments.

There are also a few known usability issues within Lark which need to be addressed. For instance, pipeline branches cannot be deleted in the current version of Lark. While this feature could be easily implemented, it represents a serious usability issue in the current system. Another usability concern that requires further investigation is conflict resolution polices for concurrent access to shared resources, a common issue in any event based system. Lark currently uses a first-come-first-serve policy, which is the most straightforward in terms of implementation, however it might not be the most usable.

Informal feedback from people suggested that interaction with Lark's pipeline meta-visualization does need to be learned and practised, as the whole interaction paradigm is quite different from familiar software. Further investigation into the possible advantages of meta-visualizations are indeed warranted, a suggestion echoed by Weaver [2005]. Furthermore, a formal evaluation of effectiveness, efficiency, and satisfaction [Gutwin and Greenberg, 2000] of the Lark system would also be appropriate. After addressing the necessary system improvement identified though such an evaluation, a

field deployment of Lark would also be an important direction for future work.

Moving to more theoretical territory, an avenue of future research could be pursued is the extension of Lark's visualization process model. Lark's model is similar to the pipeline introduced by Carpendale [1999]. As illustrated in Figure 2.7, Carpendale's pipeline could be extended to add additional resolution to the *data representation* state, potentially following the approach by Chi and Riedl [1998], separating this pipeline state into two distinct states.

Lastly, Lark's integrated meta-visualization is based on the visualization pipeline. Exploring different structuring models for the meta-visualization would be another compelling future research direction.

6.5 Thesis Conclusion

In this thesis I have presented Lark, a collaborative information visualization system that has been designed to provide active support for collaboration by setting the data-visualization within a meta-visualization. This meta-visualization is based on the information visualization pipeline and uses the levels within the pipeline to create collaboration coordination points.

My research was motivated by the benefits and requirements of face-to-face collaboration with information visualizations. With the possibility of simultaneous, concurrent interaction with a visualization on a shared large display comes the need to support the coordination of joint data analysis efforts. Lark was designed to support a range of collaboration styles by providing collaborative coordination mechanisms in an extended coordinated multiple views system. For the individual analysts this provides a new type of interaction with multiple views of the same data that are near to hand. For team work, the view structure can inform collaborators about not only what they have done, but how their work relates to what their team members have done, and the locally scoped interaction controls help coordinate collaboration. The primary benefit of Lark is an effective visual analysis mechanism for both individual and team work, help team members to switch between both types of work, and build on each others' findings.

Bibliography

Keith Andrews and Helmut Heidegger (1998). Information slices: visualising and exploring large hierarchies using cascading, semicircular discs. In *Proceedings of the IEEE Symposium on Information Visualization*, 9–12 → p. 39

Apple Inc. (2010). Cocoa – Mac OS X Technology Overview – Apple Developer. Retrieved March 20, 2010, from http://developer.apple.com/technologies/mac/cocoa.html → p. 117

Kevin Baker, Saul Greenberg, and Carl Gutwin (2002). Empirical development of a heuristic evaluation methodology for shared workspace groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 96–105. New York, NY, USA: ACM Press. doi:10.1145/587078.587093 → p. 6, 40

Michelle Q. Wang Baldonado, Allison G. Woodruff, and Allan Kuchinsky (2000). Guidelines for using multiple views in information visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, 110–119. New York, NY, USA: ACM Press. doi:10.1145/345513.345271 → p. 52, 53, 59, 62, 69, 70, 71, 74, 78, 84, 85, 87, 96

David A. Baum and Susan Offner (2008). Phylogenics and tree-thinking. *American Biology Teacher* 70(4):222–229 → p. 38, 112

Richard A. Becker and William S. Cleveland (1987). Brushing scatterplots. *Technometrics* 29(2):127–142. doi:10.2307/1269768 → p. 52

Mathilde M. Bekker, Judith S. Olson, and Gary M. Olson (1995). Analysis of gestures in face-to-face design teams provides guidance for how to use groupware in design. In *Proceedings of the Conference on Designing Interactive Systems (DIS)*, 157–166. New York, NY, USA: ACM Press. doi:10.1145/225434.225452 → p. 43

Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler (2000). GenBank. *Nucleic Acids Research* 28(1):15–18. doi:10.1093/nar/gki063 → p. 2

Sara A. Bly (1988). A use of drawing surfaces in different collaborative settings. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 250–256. New York, NY, USA: ACM Press. doi:10.1145/62266.62286 → p. 43

Susan E. Brennan, Klaus Mueller, Greg Zelinsky, IV Ramakrishnan, David S. Warren, and Arie Kaufman (2006). Toward a multi-analyst, collaborative framework for visual analytics. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, 129–136. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/VAST.2006.261439 → p. 44, 45

Ken W. Brodlie (2005). *Exploring Geovisualization*, chap. Models of Collaborative Visualization, 463–475. Kidlington, UK: Elsevier Ltd. → p. 25

Christoph Buchheim, Michael Jünger, and Sebastian Leipert (2002). Improving Walker's algorithm to run in linear time. In *Revised Papers from the 10th International Symposium on Graph Drawing (GD)*, 344–353. London, UK: Springer-Verlag. doi:10.1007/3-540-36151-0 → p. 37

Stuart K. Card and Jock D. Mackinlay (1997). The structure of the information visualization design space. In *Proceedings of the IEEE Symposium on Information Visualization*, 92–99. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/INFVIS.1997.636792 → p. 17, 25

Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman (Eds.) (1999). *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA, USA: Morgan Kaufmann → p. 4, 16, 17, 25, 28, 31, 32, 33

Marianne Sheelagh Therese Carpendale (1999). *A framework for elastic presentation space*. Ph.D. thesis, Simon Fraser University, Burnaby, BC, Canada. doi:10.1145/932254 → p. viii, xiv, 25, 28, 32, 33, 80, 88, 130

Ed H. Chi (2002). Expressiveness of the data flow and data state models in visualization systems. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, 375–378. New York, NY, USA: ACM Press. doi:10.1145/1556262.1556327 → p. 19, 25, 26

Ed H. Chi and John T. Riedl (1998). An operator interaction framework for visualization systems. In *Proceedings of the IEEE Symposium on Information Visualization*, 63–70. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/INFVIS.1998.729560 → p. xii, 15, 25, 26, 27, 28, 29, 30, 31, 32, 33, 56, 78, 100, 130

Mei C. Chuah and Steven F. Roth (2003). Visualizing common ground. In *Proceedings of the International Conference on Information Visualization (IV)*, 365–372. Los Alamitos, CA, USA: IEEE Computer Society → p. 104

Christopher Collins and Sheelagh Carpendale (2007). VisLink: revealing relationships amongst visualizations. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization)* 13(6):1192–1199. doi:10.1109/TVCG.2007.70611 → p. xiii, 55, 58, 61, 77

Christopher Mervin Collins (2010). *Interactive Visualizations of Natural Language*. Ph.D. thesis, University of Toronto, Toronto, ON, Canada → p. 32

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest (1990). *Introduction to Algorithms*. MIT Press → p. 35, 36

Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale (1993). *Human-computer interaction*. Prentice Hall International → p. 3

Selan R. dos Santos and Ken W. Brodlie (2004). Gaining understanding of multivariate and multidimensional data through visualization. *Computers and Graphics* 28(3):311–325. doi:10.1016/j.cag.2004.03.013 → p. 19, 25

David Duke, Malcolm Wallace, Rita Borgo, and Colin Runciman (2006). Fine-grained visualization pipelines and lazy functional languages. *IEEE Transactions on Visualization and Computer Graphics* 12(5):973–980. doi:10.1109/TVCG.2006.145 → p. 19, 22, 84

David J. Duke, Ken W. Brodlie, and David A. Duce (2004). Building an ontology of visualization. In *Poster Proceedings of the IEEE Conference on Visualization (VIS)*. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/VIS.2004.10 → p. 24

Scott V. Edwards and John Harshman (2008). Passeriformes: perching birds, passerine birds. Retrieved March 20, 2010, from The Tree of Life Web Project http://tolweb.org/Passeriformes/15868/2008.06.24 → p. 110

Mary Elwart-Keys, David Halonen, Marjorie Horton, Robert Kass, and Paul Scott (1990). User interface requirements for face to face groupware. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 295–301. New York, NY, USA: ACM Press. doi:10.1145/97243.97299 → p. 40, 42, 64

Encyclopedia of Life (2010a). Alaudidae. Retrieved March 20, 2010, from http://eol.org/pages/7572 → p. 110

Encyclopedia of Life (2010b). Macaca fuscata (Blyth, 1875). Retrieved March 20, 2010, from http://eol.org/pages/1037940 → p. 110

Encyclopedia of Life (2010c). Ulmus. Retrieved March 20, 2010, from http://eol.org/pages/60724 → p. 110

Free Software Foundation (1997). GNU Lesser General Public License, Version 3. Retrieved March 20, 2010, from http://www.gnu.org/licenses/lgpl.html → p. 114, 123

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley → p. 116

T. Ryan Gregory (2008). Understanding evolutionary trees. *Evolution: Education and Outreach* 1(2):121–137. doi:10.1007/s12052-008-0035-x → p. 17

Jonathan L. Gross and Jay Yellen (2006). *Graph Theory and Its Applications*. Chapman & Hall/CRC, 2nd ed. → p. 35, 36

Carl Gutwin and Saul Greenberg (1998). Design for individuals, design for groups: tradeoffs between power and workspace awareness. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 207–216. New York, NY, USA: ACM Press. doi:10.1145/289444.289495 → p. 6, 40, 48, 51, 64, 118

Carl Gutwin and Saul Greenberg (2000). 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)*, 98–103. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/ENABL.2000.883711 → p. 3, 41, 79, 102, 129

Carl Gutwin, Saul Greenberg, and Mark Roseman (1996). Workspace awareness in real-time distributed groupware: framework, widgets, and evaluation. In *People and Computers XI: Proceedings of the HCI'96*, 281–298. London, UK: Springer-Verlag → p. 43

Robert B. Haber and David A. McNabb (1990). *Visualization in Scientific Computing*, chap. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems, 74–93. Los Alamitos, CA, USA: IEEE Computer Society → p. viii, 19, 22, 24, 25, 26, 28, 33

Paul E. Haeberli (1988). ConMan: a visual programming language for interactive graphics. *ACM SIGGRAPH Computer Graphics* 22(4):103–111. doi:10.1145/378456.378494 → p. 19, 20, 107

Jefferson Y. Han (2005). Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 115–118. New York, NY, USA: ACM Press. doi:10.1145/1095034.1095054 → p. 44

Jeffrey Heer and Maneesh Agrawala (2008). Design considerations for collaborative visual analytics. *Information Visualization* 7(1):49–62. doi:10.1057/palgrave.ivs.9500167 → p. 61, 81, 85, 88

Jeffrey Heer, Stuart K. Card, and James A. Landay (2005). prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 421–430. New York, NY, USA: ACM Press. doi:10.1145/1054972.1055031 → p. 32

Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala (2008). Graphical histories for visualization: supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Symposium on Information Visualization)* 14:1189–1196. doi:10.1109/TVCG.2008.137 → p. 49

Petra Isenberg and Sheelagh Carpendale (2007). Interactive tree comparison for co-located collaborative information visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization)* 13(6):1232–1239 → p. 5, 45, 46, 47, 48, 49, 51, 52, 59, 62, 65, 66, 67, 68, 69, 71, 74, 76, 78, 84, 85, 86, 87

Petra Isenberg and Danyel Fisher (2009). Collaborative brushing and linking for co-located collaborative visual analytics of document collections. *Computer Graphics Forum (Proceedings of the Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis))* 28(3):1031–1038. doi:10.1111/j.1467-8659.2009.01444.x → p. 45

Petra Isenberg, Anthony Tang, and Sheelagh Carpendale (2008). An exploratory study of visual information analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1217–1226. ACM Press. doi:10.1145/1357054.1357245 → p. xiii, 6, 12, 41, 42, 44, 64, 71, 72, 73, 87, 120

Tobias Isenberg, André Miede, and Sheelagh Carpendale (2006). A buffer framework for supporting responsive interaction in information visualization interfaces. In *Proceedings of the International Conference on Creating, Connecting and Collaborating through Computing (C5)*, 262–269. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/C5.2006.4 → p. 107, 108, 117

T.J. Jankun-Kelly (2003). *Visualizing Visualization: A Model and Framework for Visualization Exploration*. Ph.D. thesis, University of California, Davis, Davis, CA, USA → p. 25

Paul E. Keel (2006). Collaborative visual analytics: inferring from the spatial organization and collaborative use of information. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)*, 137–144. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/VAST.2006.261415 → p. 44, 45

Russell Kruger, M. Sheelagh T. Carpendale, Stacey D. Scott, and Anthony Tang (2005). Fluid integration of rotation and translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 601–610. ACM Press. doi:10.1145/1054972.1055055 → p. 13, 74, 93, 117, 120, 129

Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Saul Greenberg (2003). How people use orientation on tables: comprehension, coordination and communication. In *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work*, 369–378. New York, NY, USA: ACM Press. doi:10.1145/958160.958219 → p. 43

Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Saul Greenberg (2004). Roles of orientation in tabletop collaboration: comprehension, coordination and communication. *Journal of Computer Supported Collaborative Work* 13(5-6):501–537. doi:10.1007/s10606-004-5062-8 → p. 41

Wolfgang Krüger, Christian-A. Bohn, Bernd Fröhlich, Heinrich Schüth, Wolfgang Strauss, and Gerold Wesche (1995). The responsive workbench: a virtual work environment. *Computer* 28(7):42–48. doi:10.1109/2.391040 → p. 41, 44

Munir Mandviwalla and Lorne Olfman (1994). What do groups need? a proposed set of generic groupware requirements. *ACM Transactions on Computer-Human Interaction (TOCHI)* 1(3):245–268. doi:10.1145/196699.196715 → p. 40, 42, 64

Kim Marriott and Peter Sbarski (2007). Compact layout of layered trees. In *Proceedings of the Australasian Conference on Computer Science*, 7–14. Darlinghurst, Australia, Australia: Australian Computer Society, Inc. → p. 37

Hassan Masum (2002). TOOL: the open opinion layer. *First Monday* 7(7) → p. 2

Tamara Munzner (1997). H3: laying out large directed graphs in 3d hyperbolic space. In *Proceedings of the IEEE Symposium on Information Visualization*, 2–10. Los Alamitos, CA, USA: IEEE Computer Society → p. 37

Tamara Munzner (2002). Guest editor's introduction: information visualization. *IEEE Computer Graphics and Applications* 22(1):20–21 → p. 17

Tamara Munzner, François Guimbretière, Serdar Tasiran, Li Zhang, and Yunhong Zhou (2003). TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility. In *Proceedings of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 453–462. New York, NY, USA: ACM Press. doi:10.1145/1201775.882291 → p. 17, 71

National Center for Biotechnology Information (2010). GenBank Growth. Retrieved March 20, 2010, from http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html → p. 3

Nokia Corporation (2010). Products — Qt – A cross-platform application and UI framework. Retrieved March 20, 2010, from http://qt.nokia.com/products → p. 117

Donald Norman (1990). *The Design of Everyday Things*. Doubleday Business → p. 31

Chris L. North and Ben Shneiderman (1997). A taxonomy of multiple window coordinations. Tech. Rep. CS-TR-3854, University of Maryland → p. 52

Chris L. North and Ben Shneiderman (2000a). Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, 128–135. New York, NY, USA: ACM Press. doi:10.1145/345513.345282 → p. 54, 61

Chris L. North and Ben Shneiderman (2000b). Snap-together visualization: can users construct and operate coordinated visualizations? *International Journal of Human-Computer Studies* 53(5):715–739. doi:10.1006/ijhc.2000.0418 → p. 54

NVIDIA Corporation (2010). GeForce 7900. Retrieved March 20, 2010, from http://www.nvidia.com/page/geforce_7900.html → p. 68

Stephen Palmer and Irvin Rock (1994). Rethinking perceptual organization: the role of uniform connectedness. *Psychonomic Bulletin and Review* 1(1):29–55 → p. 94

Tim Pattison and Matthew Phillips (2001). View coordination architecture for information visualisation. In *Proceedings of the Asia-Pacific Symposium on Visualization (APVis)*, 165–169. Darlinghurst, Australia, Australia: Australian Computer Society → p. 2, 52

Kasper Peeters (2010). tree.hh: an STL-like C++ tree class. Retrieved March 20, 2010, from http://tree.phi-sci.com/ → p. 110

David Pinelle, Carl Gutwin, and Saul Greenberg (2003). Task analysis for groupware usability evaluation: modeling shared-workspace tasks with the mechanics of collaboration. *ACM Transactions on Computer-Human Interaction (TOCHI)* 10(4):281–311. doi:10.1145/966930.966932 → p. 50

Meredith Ringel, Kathy Ryall, Chia Shen, Clifton Forlines, and Frederic Vernier (2004). Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In *Extended Abstracts of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1441–1444. New York, NY, USA: ACM Press. doi:10.1145/985921.986085 → p. 41

Jonathan C. Roberts (1998). On encouraging multiple views for visualisation. In *Proceedings of the International Conference on Information Visualization (IV)*, 8–14. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/IV.1998.694193 → p. 25, 52

Yvonne Rogers and Siân Lindley (2004). Collaborating around vertical and horizontal large interactive displays: which way is best? *Interacting with Computers* 16(6):1133–1152. doi:10.1016/j.intcom.2004.07.008 → p. 66, 86

Kathy Ryall, Clifton Forlines, Chia Shen, and Meredith Ringel Morris (2004). Exploring the effects of group size and table size on interactions with tabletop shared-display groupware. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 284–293. New York, NY, USA: ACM Press. doi:10.1145/1031607.1031654 → p. 41

Stacey D. Scott, M. Sheelagh T. Carpendale, and Stefan Habelski (2005). Storage bins: mobile storage for collaborative tabletop displays. *IEEE Computer Graphics and Applications* 25(4):58–65. doi:10.1109/MCG.2005.86 → p. 48

Stacey D. Scott, M. Sheelagh T. Carpendale, and Kori M. Inkpen (2004). Territoriality in collaborative tabletop workspaces. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 294–303. ACM Press. doi:10.1145/1031607.1031655 → p. 13, 41, 50, 73, 74, 120

Stacey D. Scott, Karen D. Grant, and Regan L. Mandryk (2003a). System guidelines for co-located, collaborative work on a tabletop display. In *Proceedings of European Conference Computer-Supported Cooperative Work (ECSCW)*, 159–178. Kluwer Academic Press → p. 5, 13, 41, 42, 43, 44, 46, 48, 50, 61, 64, 69, 73, 87, 120, 122

Stacey D. Scott, Regan L. Mandryk, and Kori M. Inkpen (2003b). Understanding children's collaborative interactions in shared environments. *Journal of Computer Assisted Learning* 19(2):220–228. doi:10.1046/j.0266-4909.2003.00022.x → p. 44

Chia Shen, Neal B. Lesh, Frédéric D. Vernier, Clifton Forlines, and Jeana Frost (2002). Sharing and building digital group histories. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, 324–333. New York, NY, USA: ACM Press. doi:10.1145/587078.587124 → p. 41

Chia Shen, Kathy Ryall, Clifton Forlines, Alan Esenther, Frederic D. Vernier, Katherine Everitt, Mike Wu, Daniel Wigdor, Meredith Ringel Morris, Mark Hancock, and Edward Tse (2006). Informing the design of direct-touch tabletops. *IEEE Computer Graphics and Applications* 26(5):36–46. doi:10.1109/MCG.2006.109 → p. 68

Chia Shen, Frédéric D. Vernier, Clifton Forlines, and Meredith Ringel (2004). DiamondSpin: an extensible toolkit for around-the-table interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 167–174. New York, NY, USA: ACM Press. doi:10.1145/985692.985714 → p. 41

Ben Shneiderman (1991). Tree visualization with Tree-maps: a 2-d space-filling approach. *ACM Transactions on Graphics* 11:92–99 → p. 37

Ben Shneiderman (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, 336–343. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/VL.1996.545307 → p. 49

Ben Shneiderman, Catherine Plaisant, Maxine S. Cohen, and Steven M. Jacobs (2009). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Boston, MA, USA: Addison-Wesley, 5th ed. → p. 52

Dave Shreiner and The Khronos OpenGL ARB Working Group (2009). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. Boston, MA, USA: Addison-Wesley, 7th ed. → p. 107, 108

Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine (2002). *The Boost Graph Library: User Guide and Reference Manual*. Boston, MA, USA: Addison-Wesley → p. 110

SMART Technologies ULC (2010). (DViT) Digital Vision Touch Technology. Retrieved March 20, 2010, from http://smarttech.com/DViT/ → p. 67

Hideyuki Suzuki and Hiroshi Kato (1995). Interaction-level support for collaborative learning: AlgoBlock—an open programming language. In *Proceedings of the International Conference on Computer Supported Collaborative Learning (CSCL)*, 349–355. Hillsdale, NJ, USA: Lawrence Erlbaum Associates Inc. → p. 43

Peter Tandler, Thorsten Prante, Christian Müller-Tomfelde, Norbert Streitz, and Ralf Steinmetz (2001). Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 11–20. New York, NY, USA: ACM Press. doi:10.1145/502348.502351 → p. 46

Anthony Tang, Melanie Tory, Barry Po, Petra Neumann, and Sheelagh Carpendale (2006). Collaborative coupling over tabletop displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1181–1290. ACM Press. doi:10.1145/1124772.1124950 → p. 6, 40, 64, 66, 86, 129

John C. Tang (1991). Findings from observational studies of collaborative work. *International Journal of Man-Machine Studies* 34(2):143–160. doi:10.1016/0020-7373(91)90039-A → p. 43, 44

James J. Thomas and Kristin A. Cook (Eds.) (2005). *Illuminating the path: the research and development agenda for visual analytics*. National Visualization and Analytics Center → p. 2, 118

Melanie K. Tory and Torsten Möller (2004). Rethinking visualization: a high-level taxonomy. In *Proceedings of the IEEE Symposium on Information Visualization*, 151–158. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/INFOVIS.2004.59 → p. xi, 17, 18

Edward Rolf Tufte (2001). *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press LLC, 2nd ed. → p. 47

Craig Upson, Thomas Faulhaber, Jr., David Kamins, David H. Laidlaw, David Schlegel, Jefrey Vroom, Robert Gurwitz, and Andries van Dam (1989). The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications* 9(4):30–42. doi:10.1109/38.31462 → p. xii, 19, 21, 22, 23, 24, 25, 28, 107

Stephen Voida, Matthew Tobiasz, Julie Stromer, Petra Isenberg, and Sheelagh Carpendale (2009). Getting practical with interactive tabletop displays: designing for dense data, "fat fingers," diverse interactions, and face-to-face collaboration. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS)*, 109–116. New York, NY, USA: ACM Press → p. 68

Matthew O. Ward (1994). XmdvTool: integrating multiple methods for visualizing multivariate data. In *Proceedings of the IEEE Symposium on Information Visualization*, 326–333. doi:10.1109/VISUAL.1994.346302 → p. 52

Colin Ware (2004). *Information Visualization: Perception for Design*. San Francisco, CA, USA: Morgan Kaufmann, 2nd ed. → p. 47

Chris Weaver (2004). Building highly-coordinated visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization*, 159–166. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/INFOVIS.2004.12 → p. 54, 61

Chris Weaver (2005). Visualizing coordination in situ. In *Proceedings of the IEEE Symposium on Information Visualization*, 165–172. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/INFOVIS.2005.38 → p. xiii, 54, 55, 56, 57, 61, 77, 96, 119, 129

Chris Weaver (2006a). Metavisual exploration and analysis of DEVise coordination in Improvise. In *Proceedings of the International Conference on Coordinated and Multiple Views in Exploratory Visualization*, 79–90. Los Alamitos, CA, USA: IEEE Computer Society. doi:10.1109/CMV.2006.14 → p. 54

Christopher Eric Weaver (2006b). *Improvise: a user interface for interactive construction of highly-coordinated visualizations*. Ph.D. thesis, University of Wisconsin at Madison, Madison, WI, USA → p. 51, 54, 70

Pierre Wellner (1993). Interacting with paper on the DigitalDesk. *Communications of the ACM* 36(7):87–96. doi:10.1145/159544.159630 → p. 41, 44

Daniel Wigdor, Chia Shen, Clifton Forlines, and Ravin Balakrishnan (2007). Perception of elementary graphical elements in tabletop and multi-surface environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 473–482. New York, NY, USA: ACM Press. doi:10.1145/1240624.1240701 → p. 43, 49, 85

The Free Encyclopedia Wikipedia (2010a). Retrieved March 20, 2010, from http://en.wikipedia.org → p. 2

The Free Encyclopedia Wikipedia (2010b). Newick format. Retrieved March 20, 2010, from http://en.wikipedia.org/w/index.php?title=Newick_format&oldid=349041266 → p. 114

The Free Encyclopedia Wikipedia (2010c). Size of Wikipedia. Retrieved March 20, 2010, from http://en.wikipedia.org/w/index.php?title=Wikipedia:Size_of_Wikipedia&oldid=347275537 → p. 3

Charles G. Willis, Brad Ruhfel, Richard B. Primack, Abraham J. Miller-Rushing, and Charles C. Davis (2008). Phylogenetic patterns of species loss in thoreau's woods are driven by climate change. *Proceedings of the National Academy of Sciences* 105(44):17029–17033. doi:10.1073/pnas.0806446105 → p. 39

Jason Wood, Helen Wright, and Ken Brodlie (1995). CSCV—computer supported collaborative visualization. In *Proceedings of BCS Displays Group International Conference on Visualization and Modelling*. Academic Press → p. 85

Richard S. Wright, Jr., Benjamin Lipchak, and Nicholas Haemel (2007). *OpenGL SuperBible: Comprehensive Tutorial and Reference*. Boston, MA, USA: Addison-Wesley, 4th ed. → p. 108

Mike Wu and Ravin Balakrishnan (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 193–202. New York, NY, USA: ACM Press. doi:10.1145/964696.964718 → p. 41

Beth Yost and Chris North (2006). The perceptual scalability of visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Symposium on Information Visualization)* 12(5):837–844. doi:10.1109/TVCG.2006.184 → p. 49

Jiaje Zhang and Donald A. Norman (1994). Representations in distributed cognitive tasks. *Cognitive Science* 18(1):87–122. doi:10.1016/0364-0213(94)90021-3 → p. 48

Appendix A

Permission for Use of Previous Publications

University of Calgary
Department of Computer Science
2500 University Drive
Calgary, Alberta, Canada
T2N 1N4

Permission for the Use of

Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale (2009). Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization)* 15(6): 1065-1072.
doi:10.1109/TVCG.2009.162

In signing this form I give Matthew Andrew Tobiasz permission to use work from the co-authored paper listed above in this thesis and to have this work microfilmed.

_____ _____

Petra Isenberg Date

University of Calgary
Department of Computer Science
2500 University Drive
Calgary, Alberta, Canada
T2N 1N4

Permission for the Use of

Matthew Tobiasz, Petra Isenberg, and Sheelagh Carpendale (2009). Lark: Coordinating co-located collaboration with information visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of the IEEE Conference on Information Visualization)* 15(6): 1065-1072.
doi:10.1109/TVCG.2009.162

In signing this form I give Matthew Andrew Tobiasz permission to use work from the co-authored paper listed above in this thesis and to have this work microfilmed.

_____ _____

Sheelagh Carpendale Date