The Pennsylvania State University
The Graduate School
Department of Geography

STRATEGIES FOR DESIGNING COORDINATED GEOGRAPHIC VISUALIZATION SOFTWARE FOR ENUMERATED DATA: A COMPONENT-BASED APPROACH

A Thesis in
Geography
by
Frank Hardisty

© 2003 Frank Hardisty

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2003
We approve the thesis of Frank Hardisty.

Date of Signature

Alan M. MacEachren  
Professor of Geography  
Thesis Advisor  
Chair of Committee

Mark Gahegan  
Professor of Geography

Guoray Cai  
Professor of Information Science and Technology

David O’Sullivan  
Professor of Geography

Deryck W. Holdsworth  
Professor of Geography  
Head of the Department of Geography
ABSTRACT

Geographic visualization software for enumerated data requires careful design if it is to fulfill its potential. This dissertation presents important problems at multiple levels of software design, and presents an approach to both design and integration of multiple EDA/ESDA methods. These design problems range from the design of coordination techniques applicable to diverse problem areas, to design of geographic visualization components for highly multivariate data spaces, to the design of what visual and numerical aspects of the data being represented should be coordinated across components for the creation of an integrated toolkit. This dissertation also presents novel approaches to these problems, and demonstrates working solutions using these approaches, at each level. Coordination among independent components (and the implementation level) and among visualization methods and tools (at the conceptual/operational level) is a major focus of the work. At the level of generic coordination strategies, the use of Java language mechanisms in combination with a standard software structure, leads to an approach that is robust and flexible. At the level of design of individual components, I introduce design ideas from Geovisualization, InfoVis, and EDA, which are implemented to create components appropriate for highly multivariate geospatial data visualization. Finally, the coordination strategies are integrated with the geographic visualization components, to form the GeoViz Toolkit, an analysis environment that is shown to provide novel insights into an important epidemiological data set.
Chapter 1

Introduction

Both the complexity of geospatial data available and the computing power that can be brought to bear on them are increasing exponentially. A challenge is to harness the increases in computer power to cope with highly multivariate data and, thus, to support understanding of the phenomena represented. Fields that analyze people in the aggregate, like population studies and epidemiology, are experiencing this challenge acutely, because electronic data collection and dissemination methods are putting more and more kinds of data at the disposal of the analyst. Phenomena associated with specific geographic locations (geospatial phenomena) are a particular challenge to analyze.

Standard methods of analyzing data about human (and other) populations do not scale well to the highly multivariate data being produced. Many of the common methods were even designed with the opposite problem in mind, that of data scarcity. Thus, new data exploration and analysis methods are required to take full advantage of increasing data availability and the increasing power of computers to process those data. Visualization often has been suggested as a tool for coping with highly multivariate data (MacEachren and Kraak 2001), through its potential to couple the power of computational methods with the power of human visual cognition to identify patterns in complex changing scenes. However, while there has been more than a decade of scientific attention to development and application of visualization methods for use in the physical sciences, engineering, and medical imaging, much less systematic attention has been devoted to developing the potential of visualization to facilitate the social sciences (Orford, Dorling et al. 1998, p. 40). While this lack of attention was highlighted in a 1998 report, the focus
on social science applications remains limited. This has caused a lack of adequate theory and practice for design of geographic visualization software for enumerated data (data collected about people in places, such as counties).

The overarching goal of this dissertation is to address a set of key research challenges about the design of geographic visualization software for enumerated data. The approach taken to meeting this goal emphasizes component-based software design. The specific sub-goals span a range from (1) software component design at the “wiring” level, to (2) the design of the visualization components themselves, to (3) the design of a coordinated toolkit.

At the system architecture level, the key goal is to find a means that allows software components to seamlessly interoperate, while the components remain open to change. At the level of visualization tool design, it is to find the means for interactively presenting large data spaces to the user. At the toolset level, it is to define a unified, but extendable, set of coordinated visual and numerical constructs that geovisualization components should support, as well as to discover which tools are better suited for aiding the analyst in visually identifying statistical and spatial patterns. Each of these goals will be explicated and met, first by describing the specific research goals addressed at each level, then describing the general design strategy which addresses the goal, and then presenting the instantiated tool (or environment of coordinated tools) which adopts the design strategy.

The implementation of these ideas is a critical step, without which the claims are difficult to evaluate. Additionally, having a functioning system to refer to strengthens our understanding of the theoretical issues. The fact that the ideas in this research have followed the path of theory to practice to the end is one of its strengths. All source code for the components discussed in this dissertation is available from (http://sourceforge.net/projects/geovistastudio/), and an executable form of the GeoViz Toolkit is available from (http://www.geovista.psu.edu/members/hardisty/).
This research has contributions to geographic problem solving in three senses (subject matter, technique, and integration with other disciplines). Firstly, the subject matter, which forms the target of analysis, is demographic, health, and related analyses of people that has been aggregated by place, i.e. geographically referenced, enumerated data. This kind of data provides an integration point for disparate data sources. These data sources can potentially provide insights into each other, but joining them together also raises some potential difficulties, which are addressed, in part, in this research. Secondly, this research draws on and integrates a number of related academic disciplines, which meet because they are applied here towards the creation of tools devoted to an analysis of places, such as counties, which are integration points for geographically referenced data. This integration illustrates both the contributions that these disciplines can have to geography, and the contribution of geography towards these disciplines.

The very act of integration of georeferenced data can produce problems, however. An analyst can easily draw together a data set that contains dozens or even hundreds of data attributes about each entity under analysis, but there is a lack of effective means for exploring relationships between such highly multivariate data sets. In this dissertation, an analysis of data is presented that draws from multiple data sources, including state cancer registries, surveys of medical facilities, telephone surveys, and census data. These data sources can be used together because they all refer to the same places. The process of data aggregation provides the raw material which multivariate statistical analyses, like multivariate regression, operate upon. However, many traditional statistical tools are not appropriate to use with highly multivariate data sets (Gahegan 2000). Beyond statistical views such regression, the joining of different data sets can enable comparisons between attribute patterns and geographic patterns. Comparisons can take the form of identifying an interesting statistical pattern, and examining the spatial patterns that correspond to it, or identifying a set of places, and examining the corresponding statistical patterns.
The second contribution to geography is to further the research agendas of some key sub-disciplines within geography. Cartography and spatial analysis have long histories of attention to methods for analysis such as classification, symbolization, and representation of geospatial data, and these areas remain ones of active concern (MacEachren, Hardisty et al. 2002). This research makes original contributions to all three of these methods. The research described here also allows these analysis methods to be applied in multiple complementary views simultaneously, increasing their power.

The third contribution to geography of this research is to use geographic visualization research problems as focal points for the integration of multiple academic disciplines. These disciplines include computer science, information visualization, and exploratory data analysis. Geographers have long investigated the use of classification and symbolization of data in georeferenced enumerated units. Expressing the insights gained from these investigations across different contexts is one means by which these disciplines are integrated in this research. Conversely, exploring the means by which other disciplines approach problems of visualizing data for enumerated units helps to arrive at novel solutions to classic problems in geographic representation.

The particular needs of geographic visualization software for enumerated data impose a number of constraints on the design of such software. These constraints guide the manner in which the architecture, component, and toolkit goals listed above are addressed. Three of the most important constraints are coordination, multi-dimensionality, and geo-awareness. The first constraint is that pieces of the software must be designed to be able to coordinate flexibly. For example, if an analyst is interested in a particular subset of the data, she should be able to easily see that subset distinguished visually in all the different data representations being used. Secondly, the software pieces should be designed to accommodate visualizing many variables at once. Maps, statistical representations, and representations that are more abstract should all be
designed with many variables, or, equivalently, many dimensions in mind. Thirdly, the geospatial nature of the data demands that the software be designed with geographic representation in mind, and that important cartographic properties, like classification and symbolization choices, should be identified and supported in a coordinated manner across all visualization components. An analyst should also be able to configure the software components to share visual properties but not data, or vice-versa.

These goals and constraints can be best addressed by applying concepts and strategies from different research domains. Four research domains are particularly relevant to developing adequate theory for geographic visualization software for enumerated data: Component and object-oriented software design, geographic visualization, exploratory data analysis, and information visualization. These areas will be introduced in the order that corresponds to the process of creating geographic visualization software for enumerated data. Computing architecture underlies the software creation process, so component and object-oriented software design comes first. Next, general techniques for representing highly dimensional demographic (and other) data need to be considered, bringing exploratory data analysis, information visualization, and geographic visualization into the picture. Geographic visualization perspectives are also used for evaluating these architectures and visual analysis techniques, using geospatial data.

Integrating perspectives from each of these disciplines has distinct theoretical and practical contributions to make to the problem domain of geographic visualization software for enumerated data, so the contributions made from examining each will be explored both separately and together. Specifically, the contribution of the four disciplines (Component-based software design, geographic visualization, information visualization, and exploratory data analysis) will be addressed in two chapters; extensions to these contributions made by this research are introduced as well. Chapter 2 focuses on computing architecture and Chapters 3 on method for exploring
highly dimensional data. These two chapters have parallel formats consisting of three steps: first, outlining how the theory behind the relevant discipline(s) applies to the problem domain and presenting research goals raised in the application of that discipline. Second, the applicability of each discipline or group of disciplines will be demonstrated by presenting a novel technique or software tool that addresses the research goal or question raised in the first step. Thirdly, the desirability of applying the domain to the problem will be illustrated by examining the advantages yielded by the novel technique or software tool.

After these two chapters covering the different disciplines, Chapter 4, then, addresses the benefits gained by using the techniques introduced in Chapters 2 and 3 in conjunction to create an integrated analysis environment, or toolkit. The development strategy for the toolkit is to adopt the coordination approaches derived from computer science, and apply them to the geospatial data visualization strategies from geographic visualization and information visualization. A proof-of-concept analysis session demonstrates the benefits of the integration.

The remainder of this chapter introduces the three levels of geographic visualization software design (architecture, component, and toolkit) and outlines the organization and specific goals of the dissertation.

Software Architecture

The overall goal for software architecture is to find a framework for software components that allows easy interoperation, combined with extensibility. This is a broad goal within which many specific goals can be identified. Two were pursued in this research:

Goal 1.1: To find ways to leverage advances in component-based software, introspection, and reflective invocation to create more stable and robust geographic visualization software for enumerated data.

Goal 1.2: To enable flexible coordination between visualization components.
A set of strategies are proposed and implemented here to address these goals regarding flexible coordination of visualization components. This set of strategies relies on component introspection, a standard naming pattern, and reflective invocation (each defined below). Component introspection is a mechanism that allows a program to query objects of which it is composed, at runtime (during program execution), to determine their available properties and methods (a method is something an object can do). The naming pattern is a standard set of relationships between software objects that can be used to match components trying to broadcast different types of messages with those that are trying to receive them. Reflective invocation is a mechanism for calling an arbitrary method in an arbitrary class at run-time, in other words, automatically manipulating the software by methods (named actions) that are discovered as the software is running, as opposed to when it was written. This runtime adaptation of the software can be applied either to the needs of the user or to those of the data. These strategies, used together, can coordinate software components, without manual user intervention, or specification of connections in source code.

The result of implementing the strategies for coordination is a working coordinator, which can be used three ways: in GeoVISTA Studio† in the GeoViz Toolkit, and on its own. The coordinator works well as a bean in Studio, and in fact was designed to ease the “wiring” burden on people assembling coordinated programs using Studio. In the GeoViz Toolkit, the coordinator performs an essential role in inter-component communication, functioning as a kind of communication clearinghouse, much as Studio does. The coordinator could also be deployed independently from all the other components mentioned in this dissertation, to coordinate other types of analysis.

---

† GeoVISTA Studio is a Java-based, cross-platform, web-deployable visual programming environment (http://www.geovistastudio.psu.edu). Studio includes a number of geographic visualization and analysis software components. The GeoViz Toolkit described in this dissertation is a sub-project within the overall Studio effort.
The structure, functions, and advantages of the coordinator can best be understood with the use of graphical diagrams that explain how the software works. The standard set of graphical definitions for object-oriented programs are contained in the Unified Modeling Language. The Unified Modeling Language (UML) can help ensure that software designs are robust in the face of changing requirements. An introduction to UML is provided, as is a summary of all eight types of UML diagrams. These eight diagram types are then used to explain the advantages of the novel coordination strategy introduced and implemented as part of the current research.

The coordinator, called the Coordination Manager, provides an automatically generated graphical user interface for changing component coordination at run-time (Figure 1-1). The coordinator, and its user interface, can be used with components having no prior knowledge of each other, as long as they follow standard Java naming conventions for message passing. Clicking on the check boxes shown in Figure 1-1 deregisters or registers one component with another for a given type of message, for example, allowing data to pass from one component to another, or to stop it flowing. The mechanism for coordination is by no means limited to data, any structure of interest can be coordinated (e.g. color schemes).
This first, more abstract contribution to the software design process has novel contributions to make in two areas, as a widely applicable software design pattern, and as a useful Figure 1-1: Automatically generated graphical user interface for coordinating software components. Checkboxes allow the user to change how components interoperate. The default, of using all available connections, is shown. The user can turn off (or back on) any specific connections between components. For example, if the checkbox under the heading “Event type: Selection” for GeoMapUni was turned off, then the component “GeoMapUni” would no longer send selection changes to the GeoMap component.
resulting tool. Generally, a software design pattern is a set of object relationships that are applicable to different contexts. Specifically, the software design pattern introduced here is to use standard naming conventions, and then to use the component-oriented software design techniques of introspection and reflective invocation of component methods to perform coordination between arbitrary components using arbitrary kinds of messages with attached software objects. This pattern could be implemented in any computer language that supports the necessary component-oriented introspection and invocation mechanisms, such as Java or C#. There is no existing pattern (known to the author) that is both as powerful and as easily implemented.

The coordinator tool I have implemented represents a direct implementation of this pattern in Java, to specifically support coordination among components designed for geospatial demographic data visualization. All of the program components described in succeeding chapters make use of this coordinator for essential coordination across (and sometimes inside) components. There is no other available single component with the same functionality (to the authors knowledge), let alone one that is available as an Open Source component, or one as small (34 kilobytes).

Geographic Visualization, Information Visualization, and Exploratory Data Analysis

A core concern shared by geographic visualization (geovisualization), information visualization (InfoVis), and of exploratory data analysis (EDA) is the process of mapping abstract data spaces onto comprehensible user spaces. The goal is to tame the difficulties of visualizing large, multivariate data spaces. That focus matches well with one of the key goals in the creation
of geospatial demographic data visualization programs: the data spaces are large, both in terms of
the number of observations, and the number of variables for each observation.

The overall goal, then, is to make highly multivariate data spaces amenable to user
interaction. Two basic strategies to address this goal are to find visualization methods that are
able to either summarize highly multivariate spaces, and make these summarizations accessible to
the user, or compactly represent multiple variables at once in an interactive way. Research goals
associated with these strategies include goals in the areas of color, data summarization, and
multivariate presentation. A substantial impediment to exploration of highly multivariate data
spaces is the complexity of multivariate relations. To overcome this impediment the following
goals and sub-goals need to be achieved:

Goal 2.1: To give analysts interactive, flexible control over data-to-color
mappings in univariate and bivariate color spaces.

Goal 2.2: To develop methods that reduce the complexity of highly multivariate
data spaces in a way that supports understanding of relationships among
variables and effective choices of which variables to explore together.

This leads us to the sub-goals:

Goal 2.2.1: To effectively present data spaces to the analyst as interactive
node-and-link graphs.

Goal 2.2.2: To use minimum spanning trees to automatically order sets
of variables in a manner that places related variables near each other.

Goal 2.3: To directly represent highly multivariate geospatial data spaces in ways
that support common classification and symbolization choices.

Thus, the sub-goals:

Goal 2.3.1: To combine pixel-based space-filling visualization
techniques with other visualization forms.

Goal 2.3.2: To develop a parallel coordinate plot that supports integration
with a common symbolization and classification system.

The results of meeting the goals for developing and implementing the visualization
strategies from geovisualization, InfoVis, and EDA are novel, independent but connectable tools,
suitable for coordination. These include: a univariate and bivariate color component suitable for use in bivariate matrices of components, and two components leveraged from other Open Source efforts, an interactive node-and-edge graph, and a PCP. The color-picking device implements a system of color anchors that can be used for data representation in conjunction with all the other visualization components mentioned here. The two components that can be inserted into matrices of other components are a bivariate geographic mapping component, and a pixel-based space-filling visualization component. These matrices of mixed representation forms support multi-way sets of bivariate comparisons. The node-and-edge graph can summarize dozens of variables at once (although only for a small number of observations), while the PCP has a unique set of directly manipulable functions that allow the user to explore highly multivariate data spaces.

Most of the components introduced here are incremental advances on similar tools developed by others – with the innovation of this dissertation focusing primarily on novel methods to coordinate the components. One component (the node-and-edge graph), is a new type of dimension-reducing visualization component that deserves separate mention. The user interface to this visualization component can be seen in Figure 1-2. In this figure, each of the contiguous 48 states (plus the District of Columbia) is shown as a node, and connections between states are shown as links. The links are determined by distance in a thirty dimensional attribute space, the attributes being composed of census variables for each state, from housing, race, income, and age variables. The links are limited to those in a minimum spanning tree (MST), a technique explained in Chapter 3. The states are colored based on the longitude of their centroids. This figure shows us that there is a strong relationship between the geographic location of each state and the total attribute space composed by the thirty census variables. Using the linkgraph coordinated with a map, it is possible to browse a large information space, looking for spatial and attribute clusters. Details of the approach taken and resulting implementation for this tool are
presented in Chapter 3 and its coordination with other components for application to demographic and health data represented in Chapter 4.

---

**Figure 1-2: Linkgraph Showing U.S. States**

---

**The GeoViz Toolkit**

Some of the core concerns of geographic visualization (geovisualization) are methods for development, implementation, and assessment of representations of highly dimensional geo-referenced data spaces (MacEachren and Kraak 2001). A coordinated toolkit, which draws from the component strategies and information visualization methods outlined in Chapters 2 and 3,
forms the example of a software implementation of geovisualization concepts, including coordinating methods for symbolization and classification of geospatial data.

Research goals that address the goal of creating an integrated geovisualization toolkit and evaluating components from it revolve around (1) delineating the visual and numerical aspects of data representation that are most important to coordinate among components and (2) designing and implementing methods to support the categories of coordination identified. These visual and numerical aspects of data representation, such as selections from a data set, or a particular categorization of the data, or a color scheme, are the essence of what is coordinated between geovisualization components. The specific goals include:

Goal 3.1: To build a typology of general categories for sets of visual and numerical aspects of data representation that should be supported for coordinated geovisualization.

Goal 3.2: To determine which specific visual and numerical aspects of representation that should populate these categories.

Goal 3.3: To give a proof-of-concept demonstration of the utility of the coordinated toolkit by exposing interesting patterns in an epidemiological data set via a coordinated analysis session.

The results of meeting these goals for creating an integrated toolkit are threefold, a typology of inter-component coordination for supporting integrated geovisualization, the GeoViz toolkit itself, and the results of a comparative analysis of various bivariate and multivariate comparisons using epidemiological data from three states in the Appalachian Cancer Network (ACN) (ACN 2003). First, the typology of inter-component coordination is a useful template of software functionality that can be applied for use in any set of visualization components, with potential applicability to other information visualization and exploratory data analysis applications. Second, the GeoViz Toolkit functions well inside or outside Studio, providing some evidence for the robustness of the design and the components. Finally, I demonstrate some of the relative advantages of the tools independently as well as what is gained by their coordinated use.
A layout of some of these tools can be seen in Figure 1-3. In this example, data on stage of diagnosis for breast cancers by county in three Appalachian states is shown. Classification, symbolization, data, and selection choices are shared by the different components, allowing the analyst to leverage the insights gained in one component in the context of perspectives provided by other components.

Figure 1-3: Coordinated toolset for geographic visualization of geospatial demographic data. Observations similar to Clearfield in high-dimensional space are highlighted across visualization components. In the Link Graph, these are the visible nodes.

While each of the four disciplines leveraged in this research (Component-based software design, InfoVis, EDA, and geovisualization) has important independent applications to the problem domain (design of geographic visualization software for enumerated data), when used in
combination, they become even more useful. The present research furnishes several examples of this cross-fertilization. In particular, the coordinated GeoViz Toolkit tools introduced in the geovisualization and InfoVis sections of Chapter 3 and applied to a real data analysis problem in Chapter 4 bear out the ideas presented in the component-based design section of Chapter 2. The result demonstrates the power of coordination among components, and a proof-of-concept analysis session illustrates this. In addition, the structure of the geovisualization and InfoVis tools can be better understood using the UML tools introduced in Chapter 2. Finally, the GeoViz Toolkit can be downloaded by interested analysts, and used with their own data.


Chapter 2

Designing Robust Demographic Data Visualization Software

Programmers spend much of their time re-implementing tasks that other people have implemented before. Why? One reason is that technologies change, and their basis becomes so radically different that it is easier to, for example, re-write a program in Java than to try and automatically translate it from Fortran, or even C++, a language in the same family. However, programmers are also perpetually re-writing code that has already been implemented in the language they are working in, even in object-oriented languages that promote reuse of code. One reason this happens is an under-acknowledged weakness of object-oriented programming: object oriented design tends to build dependencies between classes (the static definitions of objects) into a structure, making changes to classes difficult. The situation is even worse if ad-hoc design processes are followed, because the dependencies multiply and modification of a given class can cause unexpected behavior in a seemingly unrelated object, leading to a mysterious crash in the whole system.

The word “robust” is often used to describe designs and software that stand the test of time well. Robustness in software has two separate, but connected, meanings. Firstly, robust software functions correctly, without incorrect results, uncontrolled stoppage of program execution, or other unexpected behavior. This is robustness against error. Robust software can also be modified by adding or removing functions, without adversely affecting the parts of the software that were not modified. This is robustness in the face of change. Both kinds of robustness can be improved by adopting software design approaches, such as using Unified Modeling Language (UML) graphical descriptions (Jacobson, Christerson et al. 1992), and using
proven object-oriented and component-based software design patterns. A new approach for coordinating program components in the service of geospatial demographic data visualization, developed as part of the current research, serves as an important illustration of the flexibility and power of these approaches.

Programmers often lament that their software would have fewer errors if only users did not keep changing the requirements for the software. This complaint may be accurate, but it ignores the fact that requirements always change. It is impossible to determine all requirements for how software (particularly highly interactive software) should behave without seeing a version in action. Thus, the software development process has changing requirements built into it. In addition, software does not exist in isolation, but is expected to interact with users, and with other programs. Both the users and the other programs change over time, thereby changing what is required of the software. Finally, software does not spring forth fully formed, like Athena from Zeus’s head. It is developed incrementally, with errors being found at several stages. Fixing these errors requires modifying the software. If the software is not robust in the face of change, new errors will be introduced in the process of fixing old ones, and fixing the new errors will create even newer errors, and so on. If this point is reached, a system has lost its ability to grow, and it dies (Knoernschild 2002). Less dramatically, but no less fatally, if the software does not have a robust design, as the software changes over time, and as the personnel maintaining the software change, the cost of maintaining the software will exceed the cost of creating new software to perform the same functions, and the old software dies.

There is no easy solution to the dilemma of code reuse. Computer scientists are continually developing new methods that help to promote code reuse. However, for designers of software in the particular domain of geospatial demographic data visualization software, too many other specifically geographic aspects need attention to allow for use of unproven methods in the hope that they may prove productive. For these designers, it is most productive to leverage
established techniques that can be used to increase the robustness of software designs for geovisualization software. These techniques include: using UML (the Unified Modeling Language), using a component-based approach to creating software, and adopting an Open Source development model. The use of these techniques also helps address the chapter goals related to UML, the component-based coordinator, and Open Source development. The remainder of this chapter addresses these goals, in order:

- to provide a concise, but complete, guide to UML grounded in design of geospatial demographic data visualization software.

- to explain a novel and useful approach to helping software objects work together (the component-based coordinator introduced in Chapter 1), in sufficient detail such that other researchers could re-implement it in other languages, and to explain the advantages of this component over previous coordination techniques.

- to explain why an Open Source software development model is the best one for development of geospatial demographic data visualization software, especially in an academic research environment.

These goals contribute to the goals identified in Chapter 1 (Goal 1.1, leverage advances in component-based software, and Goal 1.2, enable flexible coordination) in the following ways: The first goal in this chapter, of providing a guide to UML, details a systematic methodology through which Goal 1.1 and Goal 1.2 can be reached. The second goal in this chapter, to introduce a coordinator in detail, directly fulfills Goal 1.1 and 1.2. The third goal, of explaining the use of Open Source software for improving academic software development processes, contributes to the dissemination of methods for reaching Goal 1.2.

Objects, in object-oriented languages, consist of properties and methods. Properties are the “what it has” of objects, representing a person as an object, the object might have properties like “profession”, “salary”, “colleagues”, and so on. Methods are the “what it does” of objects, continuing the same example: our person-object might have methods like “teachClass”, “goShopping”, or “meetColleagues”. These properties and methods abstractly define limits of interaction between program parts.
When writing about object-oriented software design, it is easy to think of the term “object” as referring to both classes (static, design-time definitions), and objects (run-time, instantiated instances). Some confusion between the two is understandable, because an instantiated object is always based on a class (whether on disk or resident in memory), and because some problems can be conceptualized without differentiating between them. However, keeping the two separate enables clearer thinking about both design-time and run-time issues. In this paper, following normal Java usage, “object” refers to an instantiated object, and “class” refers to the static information that defines objects. In the examples below, the object will be `geoMapOne`, and its defining class, `GeoMap`. `GeoMap` is a Java component (or `bean`) used for geographic visualization within GeoVISTA Studio.

Review of UML

UML is the standard formal, graphical, notation for specifying object-oriented software models (Gogolla, Radfelder et al. 1999). It is organized around eight diagram types, described below, that have many common elements. UML is not a visual programming language, i.e. UML diagrams are not executable. However, one can convert UML class diagrams into working code, and can even do so automatically. This “forward-engineering” capability enables the programmer to draw a diagram or a set of diagrams, and then see working code produced from it. The most popular software packages with this capability will be reviewed later in this section. These packages, as well as others, can also “reverse-engineer” UML class diagrams from existing code.

UML should be a key technique for GIScience generally, because research concerns like geographic visualization depend upon a clear representation of the entities represented. Work is ongoing to create specifically geographic extensions to UML, called Geo-UML (Bédard 1999). In this dissertation, UML helps to illustrate both the workings of the component-based coordinator
(developed in this dissertation), and the advantages it holds over a previous, object-oriented versions of a coordinator. Additionally, UML can help to focus research at the right technical level.

Given that our core concern is the design of geospatial demographic data visualization software, a question that arises is: What level of technical detail is the appropriate research focus? At one end of the technical spectrum, it could be argued that research should focus on subjects like physical memory architectures and video card drivers, because these topics form the basic substrate upon which visualization software depends. At the other end of the spectrum, it could be argued that all software issues are merely implementation details, and the core concerns should be ontological and cognitive in nature.

The middle ground, which is occupied by a range of object-oriented design and user-interface design issues, is important for all those involved in the design, creation, and analysis of visualization software for spatio-temporal demographic phenomena. This includes stakeholders such as programmers, designers of systems, those who manage the programmers and designers, and those people who represent the end users of the software (user interface experts) in the design process. Each of these stakeholders should concern themselves with different levels of abstraction in the object oriented design and software analysis process. Programmers need to be able to model individual classes and groups of interacting classes. Designers need to be able to model which software objects will contain what information and what functionality. Designers and managers need to consider how the system should work in its essential features, without regard to performance or implementation details. While the end users should not have to be aware of how the software was designed and realized, they will certainly be affected by how well the design decisions represented the needs of the end user, especially when those needs came into conflict with the needs of other stakeholders.
UML can help mediate between all these types of stakeholders. It should be used at all stages and levels of software design. Figure 2-1 helps illustrate why this is so. In this figure software design levels for object-oriented software development are shown along the bottom row. These levels range from the internal methods that objects use to do work (syntax), to the interactions between objects, to the interactions between groups of deployed objects that make up components (architecture), to the parts that the end user interacts with. Along the top, the figure shows analysis methods that can help perform each level of analysis well. The use of UML can aid all of the analytic methods listed above.

![Diagram](image)

**Figure 2-1**: UML across the software development spectrum.

The next section begins with an introduction to UML general concepts and history. Then, the different types of UML diagram will be introduced. For each diagram type, its UML elements are defined, then that diagram type is illustrated with an example drawn from geospatial demographic data visualization software developed as part of this research.

**General Concepts and History of UML**

Two important aspects of UML for the beginner to keep in mind are that different UML diagrams are alternative views of the software system under analysis, and that no UML diagram precisely defines how the system must be implemented. Different types of UML diagrams
describe the same system from different views: classes, properties, and methods versus the object states that those properties and methods embody, a user’s interactions from the perspective of the user vs. that of the machine, and so on. There are an infinite number of possible classes that correspond to a single UML diagram, just as there are an infinite number of possible valid sentences that correspond to a given valid grammatical structure. Conversely, there are a large number of ways (although not infinitely many) of describing a given class in a UML diagram, because one can exhaustively make all the valid combinations of the class’s properties and methods. However, it can be said definitively whether or not a given class is a valid representation in code of a given diagram, and whether or not a given UML diagram is a valid representation of a given piece of code. UML diagrams are like outlines of an essay; an infinite number possible essays can fit a given outline, and a given essay may be outlined in a large number of ways, but it is possible to say if a given outline legitimately summarizes a given essay, or if a given essay legitimately expands upon a given outline.

UML diagrams are summary descriptions – there is normally no need to “show all” properties and methods. In fact, the communicative aspects of UML diagrams are normally enhanced by pruning of unnecessary detail. Valid summaries may be done at many different levels of detail.

UML is a wonderful example of the success that can be achieved by compromising and unifying seemingly incommensurable theoretical approaches. Booch (Booch 1991), Jacobson (Jacobson, Christerson et al. 1992), and Rumbaugh (Rumbaugh, Blaha et al. 1991) all had different graphical approaches to representing object-oriented computer programs, but they managed to settle on a common approach, which is how UML was born. The researchers may have been aided in their efforts to compromise by the potential commercial profits to be had, as they then founded the leading vendor of UML tools, Rational Rose.
The most popular commercial programs with forward-engineering capability are Rational Rose and Together Center, and the most popular open source program with this capability is called ARGO. UML is maintained by a standards body called The Object Management Group (OMG), whose members include technology leaders such as IBM, Microsoft, and Oracle, among others. The OMG released the definition of version 1.5 of UML in March of 2003, in the *OMG Unified Modeling Language Specification* (Object Management Group 2003). In this document, the OMG lists the following eight canonical graphical diagram types, with short definitions added in Figure 2-2.

---

**Figure 2-2: UML diagram types, with short definitions**

---

The OMG organizes most of these diagram types into groups. Use cases and class diagrams are not in any group, because they involve atomic types (users and software classes) that are incorporated in other diagrams. Statechart, activity, sequence, and collaboration diagrams are behavior diagrams, which show program behavior from different views. Sequence and collaboration diagrams belong to the interaction sub-group, showing interacting classes. Component and deployment diagrams are implementation diagrams, which show how the classes are related during program execution, and how they are passed from machine to machine.

All these types of diagrams are elaborated upon below. They are introduced in the order listed above, which is the one given in the official UML specification, which in turn roughly
corresponds to the order they should be used in actual design and implementation of geospatial
demographic data visualization software. A caveat: the current official specification is 736 pages
long, and thus the description in this chapter is necessarily less detailed than the full version. The
descriptions below include every major UML diagram type, but do not describe some of the less
essential elements of some document types, in particular, less relevant parts of the definitions for
behavioral diagrams have been omitted. The descriptions concentrate on the parts of UML that
are most important for developing geospatial demographic data visualization software in a
modern, object-oriented, strongly typed language like Java or C#.

Each UML diagram type is described in some detail. For each kind of UML diagram,
first a diagram is introduced which shows the graphic elements that make up that kind of
diagram, and the elements are described. Next, an example drawn from the design and
implementation of geospatial demographic data visualization software, taken from work done as
part of the current research, illustrates how these UML elements fit together to make a diagram.

Words that are names of classes (GeoMap), or methods (selectionChanged{}), are printed
using a monospace font, and methods are suffixed with curly braces.

One application example that runs throughout this chapter (and the remaining chapters as
well) is the GeoMap component. This component is composed of several interacting classes with
a variety of relationships, as well as being an important building block for the applications
described in following chapters. Therefore, when possible, concrete illustrations will be made
using the GeoMap, and associated classes. A more complete discussion of GeoMap functionality
of is provided in Chapter 3, and its use in an integrated, coordinated toolkit is illustrated in
Chapter 4.
1. Use case diagrams – Making user needs and user interactions explicit

Use case diagrams are used to represent users of the system, and visible aspects of system behavior. They are critical for defining what the system should do. They are also the diagram type that most often contains the user’s role. Thus, they are the most important diagram type for user-centered design, which is an important area of research in geovisualization (MacEachren, Boscoe et al. 1998) (Slocum, Blok et al. 2001) as well as in other disciplines such as information retrieval (Sugar 1995; Hu, Ma et al. 1999), computer-supported cooperative work (Kies, Williges et al. 1998), and human-computer interaction (HCI) (Carroll 1997).

Elements of use case diagrams

Use case diagrams typically have three major elements: actors, use cases, and a system boundary (see Figure 2-3). These diagrams also contain connections between elements. Actors represent users of the system, normally, but not necessarily, individual people. Use cases, which appear inside system boundaries, represent classes or groups of classes that provide a coherent set of functionality. The system boundary encloses the use cases, and may be thought of as equivalent to the physical boundary formed by the hardware enclosing computational resources.
Geospatial example of a use case diagram

Considering a simple example should help explicate what use case diagrams mean (Figure 2-4). This example shows how a component builder system, such as GeoVISTA Studio, functions in terms of broad roles and functions. Here, we see three types of actors and three types of use cases. The three types of actors are component programmers, application designers, and application users. Note that the names of these actors are enclosed in guillemots. This indicates that the enclosed names are stereotypes. Stereotypes in UML are not preconceived prejudices, but rather extensions in meaning to the standard element in this case the actor element. Stereotypes are a built-in extension mechanism to UML. In this case, the stereotypes indicate that there are three different types of actors in this system, component programmers, application designers, and application users. The three types of use cases presented are providing components, assembling applications, and using applications. There are also lines between the three use cases, showing how application use depends on application assembly, which in turn depends on component creation. Lines between the actors and the use cases also show how the three types of actors match with the three use cases.

In addition, there are two notes in this diagram (Figure 2-4). Notes are free-form aspects of UML diagrams, and may be present in any kind of UML diagram. One note indicates that the
time flow is moving downwards in this diagram, with each step depending on the preceding step.

The other (meta) note identifies the first note as a note.

Figure 2-5 shows a concrete version of the three use cases in the UML diagram.

Figure 2-5 shows a concrete version of the three use cases in the UML diagram.
Use Case 1: Create Component

```java
if (layerType == DataSetForApps.SPATIAL_TYPE_
    ls = new LayerPolygon();
    ls.setOriginalSpatialData(auxData.getShapeD;
    spatialDataTypes.add(new Integer(DataSetFor.
```

Use Case 2: Assemble Application

![Studio DesignBox](Image)

Use Case 3: Use Application

![Map Interface](Image)

Figure 2-5: Use Cases in action, showing three roles that users play in the construction of applications in GeoVISTA Studio
2. Class diagrams – The building blocks

Next, we consider class diagrams, which are the workhorses of UML. Class diagrams are the most commonly used type of UML diagram. The widely used terms “object diagram” and “package diagram”, are actually not formally recognized diagram types, but instances of class diagrams with certain features. Object diagrams, then, are class diagrams showing run-time objects. Package diagrams show all classes that appear in the same package.

Elements of class diagrams

The compositional elements of class diagrams are depicted below in Figure 2-6. These elements appear in most other types of UML diagrams, and thus are worth understanding clearly. The first three elements are the “class”, “object”, and “interface” elements. All of these together are referred to as “classifiers” in the UML specification, but this usage is highly confusing in the context of geospatial demographic data visualization software (where a classifier is a method for grouping data into categories), and the term “classifiers” will not be used in this document. Instead, each element will be referred to separately.

---

Figure 2-6: Elements of Class diagrams
Classes are the static structure that describe objects. Objects are, in turn, instances, or instantiations, of classes. Classes are the blueprints; objects are the houses. However, this is not a metaphor that should be pushed too far, as in Java, for instance, where every class can be treated as an object. You can treat a class as an object, but you cannot live in a blueprint!

Interface is an overloaded term in computer science. It has three important meanings in the context of this research: the user interface, “class signature” interfaces, and interfaces that act like classes. The user interface links the user to the system and is most often a graphical user interface, consisting of menus, toolbars, buttons, etc. The next two meanings of the term interface (interfaces that are class signatures and interfaces that are classes) are related but distinct. An interface in the “class signature” sense is the total of all the properties and methods that are visible, or accessible, from outside a class. It is the sum total of the ways that other classes may interact with a given class. This includes data types and other classes that can be returned by the methods of a class. For example, a GeoMap can return the number of layers it has, the current background color, or the set of spatial data it is rendering as a map.

Interfaces that act like classes, in Java, are a type of class that has no body to its methods; in other words, it consists of an abstract classes’ “class signature” without any implementation (an abstract class being a class which cannot be instantiated). Interfaces that act like classes are used to define roles that other classes may play. If a class implements an interface, that means it is prepared to be referred to as being that type. These interface roles are similar to the roles people play; we may act as teachers, students, parents, and sons or daughters, all within the course of a day. It could be said that a person who is able to act these roles implements the teacher interface, the student interface, etc. Thus, if it is known that a particular person is a teacher, then they may be expected to have the attributes of a typical teacher (has students, has courses), and be able to respond to the typical types of requests that a teacher responds to (give lectures, grade papers).

Similarly, in Java, if a class implements the interface SelectionListener, which has a
single method `selectionChanged()`, we know that we can call the method `selectionChanged()` on this class. The class plays the role of a selection listener, which means that it is listening for selections.

Interfaces in UML are represented by a graphical notation that looks like a lollypop, with a circle on the end of a stick (Figure 2-6, top right). Since interfaces are really classes, we can think of these lollipops as shorthand versions of class diagrams. If an interface is represented as a class diagram, the convention is to use italics for the class name, to indicate its abstract status.

Next, we consider the various types of association between classes that can be represented in UML. The most general type of association is a line drawn between classes. This implies some sort of relationship between them. It is much more useful if the association has some additional information (indicated by having a particular end type), and so all the examples we show here will be of this kind. A directional association represents a reference from the class at the base of the arrow, to an object instantiated from the class at the tip of the arrow (Figure 2-6, bottom left). If the line is dashed, this is called a “dependence”, perhaps somewhat confusingly, because all types of association between classes imply some kind of dependence. This dependence arrow tells us that there is a reference in the class at the base of the arrow to static, class level information about the class at the tip of the arrow (Figure 2-6, bottom, second from the right). Normally, these kinds of references are necessary for one class to call another classes’ methods.

Extension and implementation (Figure 2-6, bottom middle) are stronger and more specific types of association. They imply an identity relation between two classes. A child class that extends a parent class assumes all the constructors, properties, and methods of the parent class. In Java, a class may extend only one other class. At the same time, Java enables a class to implement any number of interfaces. In other words, a class can have only a single fixed set of
characteristics, but it can take on many different roles. Extending concrete classes is called “implementation inheritance”. Extension uses a solid line; implementation uses a dashed one.

Aggregation and composition are also strong types of association (Figure 2-6, bottom right). An open diamond at the base of an aggregation arrow shows that the class at the base is partly composed of the class at the tip. So, for instance, a map might have a tooltip as one of its constituent parts, so we would draw an aggregation arrow from the map class to the tooltip class. Composition is a specific type of aggregation; it means that the class at the base of the arrow is exclusively responsible for the creation and destruction of the class at the tip, and that there is no shared reference to the class at the tip from a third class. Therefore, in the preceding example, we could not draw a composition arrow from the map to the tooltip if the tooltip may be shared with other classes. On the other hand, if the map had a drawing area that it alone was responsible for, it would be appropriate to draw a composition arrow.

Geospatial example of a class diagram

All these types of elements should become clearer with an example diagram to refer to. First, we examine the use of the UML elements that show relationships between classes, and then show how UML may be used to represent the composition of classes. The example diagram in Figures 2-7 and 2-8 illustrates all of the specific types of association between classes shown in Figure 2-6. All of these relationships can be referred to as being from GeoMap to some other class. Figure 2-7 shows the user interface for the class GeoMap, and some of the associated components that GeoMap is composed of. This figure is not intended to show the class relationships, but to concretize the discussion (the class relations are shown in Figure 2-8).

There are two associations of the association or dependence type. GeoMap is associated with the ColorSymbolizer class, and depends on the SelectionEvent class. What do these
associations mean? The association with ColorSymbolizer shows that there is a reference to an instantiated ColorSymbolizer object within the GeoMap class. Therefore, any changes to the ColorSymbolizer class will be reflected in any instance of the GeoMap class. The dependence on SelectionEvent shows that there is a reference to the SelectionEvent class within the GeoMap class, but there is no reference to an instantiated SelectionEvent object in GeoMap. Thus, the line connecting the SelectionEvent to the GeoMap is dashed instead of solid. The dependence relation is more specific, and more limited, than the association relation is, because it excludes the possibility of dependence on an instantiated object based on the class.

Figure 2-7: User interface for the class GeoMap, with associated classes
Figure 2-8 shows that GeoMap has three associations that are forms of inheritance, one extension, and two implementations. GeoMap extends JPanel, indicated by the solid line with a triangle at the end, and implements SpatialExtentListener and SelectionListener indicated by the dashed lines with triangles at the end. These associations tell us that all the methods in JPanel, SpatialExtentListener, and SelectionListener can be called on in GeoMap. Further, we can ask GeoMap to be represented as a JPanel, SpatialExtentListener, or SelectionListener. The implications in terms of code stability are quite different, though. The fact that GeoMap extends JPanel means that any changes to the methods of JPanel or any of its superclasses will be reflected in the behavior of GeoMap, unless the methods are overridden in GeoMap. On the other hand, since interfaces have no implementation, unexpected behavioral changes in GeoMap will not occur because of changes to the interfaces SpatialExtentListener or SelectionListener. All of
the implementation code to support these interfaces resides in GeoMap. This guarantee of
stability has a disadvantage, though in that these interfaces cannot be extended without also
extending GeoMap while JPanel may be extended, without changing the source code of GeoMap.

Next, we consider the aggregation and composition relationships. Figure 2-8 shows one
aggregation relationship with the Fisheyes class (a class that supports distorting the map as if
using a fisheye lens), and composition relationships with MapCanvas and VisualClassifier. This
means that Fisheyes forms a part of GeoMap. One test for whether the aggregation modifier is
justified is to try putting the classes in a sentence, connecting them with “has a”, and seeing if it
makes sense, in the context of how the classes actually interact. In this instance, to say “GeoMap
has a Fisheyes” is an appropriate description of the relationship, so we draw an aggregation
association. SelectionEvent is used by GeoMap, but to say “GeoMap has a
SelectionEvent” is not accurate, because GeoMap passes a SelectionEvent along to
the MapCanvas for rendering, but does not manipulate it or maintain a reference to it. This
difference arises because the selections can come in from outside, but the fisheye is particular to
the map. The MapCanvas itself handles selections that are generated by the user in the map.

Figure 2-8 shows two composition relationships between GeoMap and other classes,
MapCanvas and VisualClassifier. Further, it shows the number of the instantiated
objects that the relations refer to. The composition lines, and the numbers next to the arrowheads,
show that an instantiated GeoMap has exclusive access to one instance of MapCanvas and two
instances of VisualClassifier, and that GeoMap is responsible for their creation and their
disposal. The line and numbers connecting MapCanvas to ShapeLayer show that a
MapCanvas has 0 to N ShapeLayer objects, and that a VisualClassifier is made up of
exactly one ClassifierPicker and one ColorRampPicker.
Example of different levels of detail in class diagrams

Extending from the above introduction, we need to consider how we may represent more detailed information about classes. Figure 2-9 shows three alternative representations of the GeoMap class. All of these are valid representations of the same class. The representation with the least detail is the summary level diagram, which is the level of detail that has been used in the previous figures. The next, more detailed level is an Analysis Level Class Diagram, which shows two additional compartments, separated by horizontal lines: a compartment for member variables and one for methods. Implementation level diagrams are most useful for considering the structure of a single class, which the other two are more useful for considering the relations between classes, and will be described in more detail below.
In the analysis level class diagram, members are represented in the form “member name : type”. Therefore, in the case of the first member listed in the analysis level class diagram in Figure 2-9, we have `mapCan : MapCanvas`, `mapCan` being the name of the object, and `MapCanvas` being its type, in this case a class. Types in Java are either classes or primitive data types like `int` or `boolean`. Methods are represented in the form “method name(argument type) : return type”. The first method listed in the analysis level diagram in Figure 2-9 is “getColors() : Color(Knoernschild 2002)”, so the method name is “getColors”,

![Figure 2-9: Different class diagrams at different implementation levels, representing the same Java class](image-url)
the argument type has been omitted, and the return type is an array of Colors. Typically, in an analysis level diagram, not all members and methods would be shown, just the ones most relevant to the diagram being created. Members and methods can, and should, have “visibility markers” attached to them. These are plus or minus marks next to the name of the member or method. The plus mark indicates that the member or method is public, and therefore visible to any class which has access to GeoMap, while a minus indicates that the member or method is private, and only visible inside the class.

An implementation-level class diagram can include arbitrarily more information about the class being represented. In the example given in Figure 2-9, the implementation level representation of GeoMap shows the argument type for all methods, and has separate sections for all interfaces implemented by this class, as well as separate sections for private members and for those with accessor methods, and for event types that the class can broadcast. The implementation-level class diagram shown was automatically generated from source code using TogetherCenter. Additionally, some implementation level diagrams include default values for members, or embed class diagrams within the class to show more detail about member types.

3. Statechart diagram – Program behavior

Statechart diagrams describe the behavior of State Machines (Harel 1987; Harel 1997). State machines generally are one kind of view for representing computation; they can be used to completely represent arbitrarily complex programs (Gurevich 1994). Specifically, statechart diagrams describe the states that a computational element like a class may be in, and the triggers that can cause transitions between states. They are especially useful for describing the behavior of components that react to user input.
Elements of a statechart diagram

The elements of statechart diagrams can be seen in Figure 2-10. From left to right in Figure 2-10, they are a start state, and end state, an intermediate state, and events that show transitions between states. The start state and the end state are atomic; they may not contain any other elements. The intermediate state will contain classes and events. The intermediate state may also contain the start state, the end state, or more intermediate states, i.e. intermediate states may be nested. The events are the elements that show changes between states, and they are generally labeled with the condition that causes the change.

![Diagram of statechart elements](image)

Figure 2-10: Elements of a statechart diagram

Geospatial example of a statechart diagram

These elements can be seen in use in Figure 2-11. This figure shows the state of the tooltip in a MapCanvas. The default state is an idle one. The transition to the active state happens when a MouseEnter event is received by the MapCanvas. The active state has sub-states, shown inside it. The initial state is triggered each time the active state is entered. This leads the tooltip to enter the state labeled “Show Default Tooltip”. If there is a pause in the motion of the mouse of more than some arbitrary length of time (one second in the example), the tooltip enters the “Show Excentric Tooltip” state. The tooltip will remain in one of these states until a MouseExit event is received, at which time the tooltip goes into the idle state.
4. Activity diagram – Program flow

Activity diagrams are similar to statechart diagrams; they both show program behavior as a result of different conditions. They differ in that the statechart is focused around states, while the activity diagram is focused on the decision branches. An activity diagram shows how a process goes through various states based on internal program flow; it represents the process of computation taking place. These diagrams are easy for UML novices to interpret, because they resemble traditional flowcharts. They are especially useful to describe branching program control flow.

Elements of an activity diagram

The elements of activity diagrams are shown in Figure 2-12. Some of the elements of an activity diagram are shared with the statechart diagram; the start and end states are identical. The action state is analogous to, but different from, the “state” element of the statechart diagram,
while the decision state is included only in activity diagrams. The action state is something that is to be done, like a task on a “to do” list. A decision is a place where control branches, in the same manner as a traditional flowchart.

![Elements of an Activity Diagram](image)

Figure 2-12: Elements of an Activity Diagram

*Geospatial example of an activity diagram*

Figure 2-13 illustrates an activity diagram that represents how a user's change in the number of classifications in a GeoMap propagates through the different parts of a GeoMap. This activity starts when the user changes the slider’s state inside the VisualClassifier. There is then a decision: Is the slider’s current value different from the previously recorded state? If not, nothing further is done, and we transition to the end state. If the value is different, we pass to the GeoMap, where a bivariate scheme is constructed. Control then passes to the MapCanvas, where the colors that correspond to the scheme are found, and passed to the ShapeLayer, if one exists. The ShapeLayer then updates its colors and repaints itself.
5. Sequence diagram – As time goes down

Sequence diagrams are different from other UML diagrams in that they show how actions and interactions happen between classes using a uniform timeline for all classes and actors. The timeline has a few features that are worth highlighting. Firstly, the timeline always proceeds from top to bottom, with earliest time on top, and later time on the bottom. Secondly, the timeline is not guaranteed to have a uniform scale, it just guarantees the sequence of actions, i.e. it is ordinal and not metric or numeric.

Elements of a sequence diagram

Figure 2-14 shows the basic elements of a sequence diagram. Sequence diagrams may include actors, as in use case diagrams. Normally, the classes in sequence diagrams are represented as objects, because they must have been instantiated. A dashed line indicates the
lifeline of the actors and classes, while rectangles placed on the lifelines indicate the start and end of activation for the particular events being represented. There are three types of calls between objects: messages, procedure calls, and returns from calls. A message is a notification that something has happened. A procedure call is a call on some method. The procedure to call might be attached to a message. The return from call indicates what is returned from a procedure call. All names of messages, procedure calls, and returns from calls should be indicated with stereotypes, to identify them for the diagram reader.

![Figure 2-14: Elements of a Sequence Diagram](image)

Geospatial example of a sequence diagram

To illustrate how sequence diagrams work, we will use the same overall example that was used for activity diagrams, of a classification being changed in a GeoMap. Although the software objects and actions being represented in the two kinds of diagram are the same, the two diagrams are quite different (Figure 2-15). This helps the program analyst focus on different aspects of the operation. The activity diagram helps the analyst see the control points in the process: slider changes, scheme finding, error states, etc. The sequence diagram shows the time element at work: what happens first, what happens next, and so on. In the case of Figure 2-15, the diagram shows more clearly that the sequence of events starts with the user, passes to the VisualClassifier, which interacts with the GeoMap to classify the data. This information
is then passed to the `MapCanvas`, and from there to a `ShapeLayer` to perform the actual changes.

---

**Figure 2-15**: Example of a Sequence Diagram, showing the sequence of events when a user changes the number of classes in a GeoMap

### 6. Collaboration diagram – Classes working together

Collaboration diagrams generally depict collaborations between classes, not between people, or between people and classes, although it is possible to include actors in them.

Collaboration diagrams are closely related to activity and sequence diagrams, in that they all show classes working together over time to solve some problem. The difference between a sequence diagram and a collaboration diagram is that collaboration diagrams have numbered messages rather than a timeline to illustrate the sequence of events. This makes collaboration
diagrams better suited for depicting parallel processes that diverge and merge, or iterative processes that repeat over time.

**Elements of a collaboration diagram**

Figure 2-16 illustrates the elements of a collaboration diagram. Actors are the same as presented in use cases, objects and associations are the same as in class diagrams. Messages are a new element; they contain time sequence information as well as the meaning of the message passing between objects. Collaboration diagrams can better show the context of interactions between classes, compared with activity diagrams and sequence diagrams. These diagrams can also better illustrate how a class may take on different roles during interactions between classes.

![Collaboration Diagram Elements](image)

**Figure 2-16: Elements of a Collaboration Diagram**

**Geospatial example of a collaboration diagram**

Figure 2-17 illustrates a collaboration diagram, using the same example of a user changing the classification in a GeoMap. In comparison with an activity diagram, the collaboration diagram better captures the messages passing between classes and the roles that the classes take on. The activity diagram in turn better captures the decision points and possible changes in program flow.
7. Component diagram – Runtime arrangement and communication

The last two types of UML diagrams use components. In the words of one of the experts in the field, “…software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system.” (Szyperski 1999). Traditional systems are “fully integrated”, they must be compiled together and then form one binary whole. A single class cannot be changed in such a system. In component systems, by contrast, the components that make up the system may be changed without breaking the whole. As an example, think of the “file picker” that opens up when a user tries to open a file in Microsoft Word, or Excel, or any number of other programs. This file picker looks the same in different programs because it is the same; on the computer’s hard drive, there is only one binary blueprint, which becomes instantiated and configured on demand. Similarly, a Visual Basic component that uses the interface for the system file picker can work on different Microsoft platforms, such as Windows 95, 98, NT, or XP, and will open whichever file picker is resident in the system (and thus it is able to take on the role of different file pickers).
Elements of a component diagram

Figure 2-18 shows the elements of component diagrams. The same diagram elements represent interfaces and associations here as used in class diagrams in general. Recall that interfaces can be thought of as compact forms of class diagrams, as interfaces are classes. The novel element in this kind of diagram is the component itself, which should be identified by name, and optionally given a role by a stereotype. Components are deployed binary code, as described above.

Geospatial example of a component diagram

Figure 2-19 illustrates how a component diagram may be used. The particular case shown is that of a geographic data set being read by the GeoDataReader component and then passed to a ShapeFileProjection component, then to a ShapeFileToShape, for transformation from the proprietary shapefile (.shp) format to more generic Java Shape format. Finally, the GeoMap receives the data, via the DatasetListener interface that it implements. This diagram shows some undesirable dependencies on particular components on the part of GeoDataReader and ShapeFileProjection, and that ShapeFileToShape has a more desirable, abstract dependency on the DatasetListener class instead of depending on GeoMap directly. Abstract dependencies are more desirable because they allows the class being depended on to be changed without changing the class with the dependency. In this case, it means
that other classes that are data providers, such as a database, can supply the data without changing GeoMap.

---

**Figure 2-19: UML Component Diagram showing data flow from a file reader to the GeoMap**

---

### 8. Deployment diagram – From one computer to another

Deployment diagrams illustrate how components pass from physical place to physical place, and how components are composed and connected. They share many elements with component diagrams.

#### Elements of deployment diagrams

Deployment diagrams contain an additional “node” element, as well as the elements that a component diagram has, as can be seen in Figure 2-20. Nodes are physical computational resources, with at least storage facilities and often processing facilities as well. Nodes may be human resources.
An example of a deployment diagram in use can be seen in Figure 2-21. This diagram shows the deployment of GeoVISTA Studio, starting from source code, and ending with the delivered application. We start with the classes, in source code, on the Studio project web site at Sourceforge, which hosts Open Source projects (the meaning of Open Source is described later in this chapter). The source files are compiled into class files and grouped together into Jar files, which are the standard Java files for deploying class files, and placed on a server at the GeoVISTA Center. Then, when the client accesses the GeoVISTA web site, the components are downloaded to the client’s location, and instantiated there.
Object Orientation vs. Component Orientation for Coordination of Geovisualization

This section outlines the distinction between object-oriented and component-based software. Then, in two sub-sections, it introduces the challenges associated with coordination among geovisualization analysis tools and current implementations and uses of coordination. The remainder of the section (six subsections) presents a detailed discussion of coordination for geovisualization as addressed in this dissertation. UML is used to clarify the discussion in the later subsections.

Object-oriented vs. Component-based programming paradigms

NATO sponsored a conference in 1968 (Dijkstra 1968) to address the “software crisis”, i.e., computer hardware systems were increasing in power and complexity faster than the software systems that made them work. Programmers could not reuse computer code from one system to another, resulting in a great deal of repeated effort and repeated mistakes. If this “crisis” sounds familiar, it is because the disconnect (between hardware power increasing ever faster while software power does not) has not been resolved in the intervening 35 years. However, incremental improvements in code maintainability and reusability have been achieved as new programming paradigms have emerged, made possible by increasing computing power.

The most recent shift in programming paradigms adopted by commercial software vendors has been the change from object-oriented to component-based programming (Adler 1995; Szyperski 1998). Object-oriented programming relies on combining state and behavior in constructs called classes. Component-based programming relies on interactions of runtime, binary constructs. There is a good deal of overlap between component-based programming and object-oriented programming, because components are most often written using an object-oriented
programming language. Component-based programming addresses some of the deficiencies of
object-oriented programming, just as object-oriented programming addresses some of the
shortcomings of procedural programming. The goal of each programming paradigm change has
been to promote re-use of code.

People were often unable to take advantage of previously written libraries of procedural
code because of a lack of encapsulation. Object-oriented programming techniques allow for better
namespace management and encapsulation of data and procedures (Vanoosterom and Vandenbos
1989). Namespace management and encapsulation both refer to means for limiting interaction
between parts of software, namespace management at a global level, and encapsulation at the
local level. Namespace management is based on the concept of the namespace, which is the set of
named programming constructs that are accessible from different points of program execution. In
Java, for example, a class has access to all classes in its package, and the java.lang package,
which contains the basic classes of the Java language. All other classes must be explicitly
imported to be used. Different classes thus have different sets of classes that they can access. This
differential access is the heart of namespace management. The superior namespace management
of object oriented systems overcomes one of the critical weaknesses of large procedural
programs, that there was only one global namespace. This lead to increased program fragility,
because changes to any global constructs could have unexpected consequences elsewhere.
Encapsulation complements namespace management by making it possible to show or hide parts
of classes. The two most important classes of accessibility are public and private. A public
property can be accessed by any class that has access to the class with that property; a private
property can only be accessed by the class holding that property. Encapsulation allows a class to
insulate its operations from changes elsewhere in the program.

Using a procedural programming language, each variable defined may interfere with
others defined elsewhere, while object-oriented programming limits how widely a variable may
be “seen”. However, this leaves us with the problem that these relationships need to be defined at
design time (when the program is written), instead of at run time (when the program is used by an
analyst). Many relationships are pre-defined in source code, making them inflexible (except the
programmer with access to the source).

Component-based programming tries to mitigate the problems above by allowing the
programmer to create interactions between computing elements that depend not on some specific
object behaviors but on a binary contract of what will be passed between the elements. This
enables run-time changes to be initiated by a user who does not have access to (or perhaps even
the skills to change) the source code.

Why Coordination?

Developing mechanisms to support coordination among data analysis tools addresses a
fundamental challenge in visually-enabled geospatial information analysis, that of finding a
means to allows the analyst to construct an analysis environment suited to their particular analytic
needs. Typical information analysis and visualization tools currently available often show a lack
of coordination among the windows of multi-window displays, as discussed by North and
Schneiderman (North and Shneiderman 2000; North 2001), and Roberts (Roberts 1999). The
problem, however, extends beyond the lack of support for coordinated views to lack of support
for dynamic coordination among tools (e.g., a spreadsheet, a statistics package, a map). This is a
direct consequence of designing tools that operate within a closed-world, having poor
mechanisms for inter-tool connections. The resulting lack of coordination constrains analysis
activities because of the conceptual and practical barriers faced when moving the focus of
analysis between tools. Consequently, tools tend to be used in linear sequence, and the tools do
not leverage the information added by the user’s actions across different programs. For example,
if a user classifies and chooses a symbolization that reflects that classification in a map package, a loosely linked environment will not allow that classification and symbolization to be automatically reflected in a statistical package.

Most previous work on coordinated views has focused on coordination of selection (brushing points on one scatter plot highlights the same entities on all linked scatter plots, or on other linked views). For visual data analysis, however, coordination of other user actions is also needed. Most maps of numerical data apply classifications of data into 3-7 categories defined by data ranges and the resulting classes are depicted with common symbolization (e.g., a common color or symbol size). Selection can be seen as a special case of classification, into two classes (selected and not selected). Whether entities are grouped into classes by direct manipulation (e.g., brushing), or by indirect application of classification rules (e.g., dividing ranked values into quantiles having the same number of values in each class), dynamic coordination across views as classification is manipulated can enable insights into complex data relationships. In addition to coordinating classification actions by the user, it can be equally useful to coordinate user manipulation of symbolization changes applied to one classification (e.g., changing the color scheme used to represent classes in one view is reflected with a similar or identical color scheme change in other views).

As an example of coordinated selection, classification and symbolization, consider an epidemiologist who is examining disease patterns using three different views, one numerical (a spreadsheet), one statistical graphic (a parallel coordinate plot), and one geographic (a map). Using traditional linking and brushing techniques, she might be able to identify relationships between the geography and the attribute data. Using selection in the PCP, she might be able to see that areas with high values in one kind of cancer are low in a different kind of cancer. If this is linked to a map, she might see that these high values are clustered in space. Conversely, using selection in a map, she might be able to see whether a set of contiguous areas have high or low
values, using the PCP. However, what if the epidemiologist is interested in multi-category patterns? She might divide the observations into low, medium, and high values, and color them accordingly in the map. Then, the map, the PCP, and the spreadsheet should all express these categories. See Figure 2-22 for an example of such coordinated classification and selection. A longer worked example of the advantages of coordination is provided in Chapter 4.

![Figure 2-22: Coordinated Classification and Selection with a table, a map, and a PCP, with Toole County in Montana selected](image)

When trying to construct useful knowledge (such as the most effective categorical schema that could be developed for use in the mapping example above), it is commonplace to move back and forth between analysis activities: data are explored, hypotheses are created, categories are constructed, analysis is carried out, and results are evaluated, e.g. (Fayyad, Piatetsky-Shapiro et al. 1996). To be successful, these various analytic tasks need to have a high degree of connection,
which implies that the tools that enable the tasks should be coordinated. If poorly coordinated tools separate evaluation from problem formulation, or separate visual data exploration from model application, then this separation hinders the scientist who is trying to address these many inter-related stages concurrently. Thus, a coordinated environment is needed, where these various stages of science can be conducted in parallel or indeed in any order, and with minimum inertia that might prevent the researcher from moving between them, as illustrated in Figure 2-23.
Current Implementations and Uses of Coordination

Information visualization and EDA strategies rely increasingly on use of two or more software applications to achieve desired functionality. Recent examples include the linking of ArcView® with XGobi (Cook, Symanzik et al. 1997), Snap-together Visualization (North and Shneiderman 2000), and Orca (Sutherland, Rossini et al. 2000). To be fully effective, these systems need to support cross-application object selection with linking, and a consistent appearance, as well as linking of statistical attributes (Unwin 1999). This means, for example,
that it should be possible to examine data in a map, a parallel coordinate plot, and a spreadsheet, while linking important visualization behaviors across program components, even when these components were developed independently.

“Linking and brushing” (Joining different views on a set of information by visually highlighting the same objects on different views) can be traced back to (Fishekerkeller, Friedman et al. 1974) and (Newton 1978). Initial applications to geospatial data were proposed by Carr (Carr, Littlefield et al. 1987) and Monmonier (Monmonier 1989) and applied subsequently in many applications, e.g., (Cook, Majure et al. 1996; Dykes 1997; Andrienko and Andrienko 1999; MacEachren and Kraak 1999). These studies provide evidence that coordinated views for information visualization and query (focused on standard linking and brushing) that coordinated multi-view environments are effective tools for access and analysis of complex information.

Strategies for visual coordination across applications (as well as views) beyond linking and brushing have been implemented using a pipeline metaphor in several scientific visualization packages (e.g. AVS, IBM Data Explorer). Figure 2-24 illustrates the pipeline graphic interfaces provided by AVS and Explorer. Coordination of program components by means of pipelines can be considered a form of (visual) programming, especially when the resulting program state can be stored independently, as in the AVS system.
The assumption underlying all visual-programming environments for visualization is that providing linkage of program components via visual programming will enable both the rapid creation of novel visualization methods, and increase the accessibility of advanced geovisualization for non-programmers. Scientific visualization environments using this pipeline-based visual programming approach have been applied frequently to exploration of geospatial data (Trenish 1995; Wood, Wright et al. 1997; Gahegan 1998; Masters and Edsall 2000). Within scientific visualization, the focus of work on coordination has emphasized development of support for multi-user environments (Brodlie, Duce et al. 1998; Watson 2001). Brodlie (Brodlie in press) identifies three architectural models through which multi-user coordinated visualization can be achieved: a single shared application, a single replicated application, and multiple, independent applications, interlinked as a single, distributed application.

A related project, which extends the notion of object-orientation into “actor-oriented” programming, is Ptolemy (Liu, Eker et al. 2003). Ptolemy allows for visual programming, just as Studio does. In addition, in common with Studio, it uses reflection, and supports a wide variety of computing paradigms. However, it is time-based, rather than event based. In addition, Ptolemy is

Figure 2-24: Interfaces for AVS (left) and IRIS Explorer (right)
directed at creating real-time systems, rather than visualization environments. Figure 2-25 illustrates what a Ptolemy user interface looks like.

Figure 2-25: Ptolemy user interface (Liu, Eker et al. 2003)

In the section below, an approach to extending the potential for coordination among visual and computational components of analysis is outlined. The approach is being implemented within a Java-based visual programming environment (GeoVISTA Studio). GeoVISTA Studio provides rapid, programming free development of complex applications for data exploration, knowledge construction, geocomputation, and visualization (Gahegan, Takatsuka et al. 2001). It forms an ideal platform for advancing the state of ESDA, and provides an easy delivery mechanism for geospatial data analysts.

**Studio as a coordinator**

GeoVISTA Studio itself has coordination as one of its major features. It allows fine-grained control and almost limitless flexibility in how components can be connected, and therefore coordinated. What, then, is the need for an additional coordinator? One reason is to reduce the burdens on the design maker (the person constructing designs in Studio, as shown in Figure 2-4). If we have 15 components in a design, and each one can coordinate in 6 ways, then
we will need $15 \times 6 \times 2 = 180$ links for bi-directional association. Considering that in Studio one must use a GUI and make choices (via a wizard) upon making each link, the process would be tedious and time consuming. It is more practical to add an additional coordinating component that does some of the wiring automatically, and hide the complexity of multiple inter-connections from the program developer by using a single coordinator connection. Figure 2-26 illustrates the degree to which the coordinator simplifies a design, even one with only a few components. Since the number of connections expands exponentially with the number of coordinated components, this advantage is even greater for designs with many components. The functionality in the two coordinators described below does not replace the functions of Studio, it complements them. Studio was also the intellectual forerunner of both.

Figure 2-26: Studio designs without (left) and with (right) coordinator bean

**An Object-Oriented Coordinator – GvCoordinator**

The basic design of the first, object-oriented coordinator used in Studio was that of a multi-caster; Coordination events were fired by objects being coordinated, intercepted by the
coordinator, and then sent by the coordinator to all listening objects. Figure 2-27 illustrates the simple process that takes place when a component is added to the object-oriented GvCoordinator. The bean is added to a list of beans that are being coordinated. One consequence of using one master list for all coordination is that as this list grows, it needs to be traversed an increasing number of times as the number of components and types of coordinated events grows.

Figure 2-27: UML activity diagram showing how a bean is added to the object-oriented GvCoordinator

Figure 2-28 shows the sequence that occurs when two components, a GeoMap and a Scatterplot, are added to the GvCoordinator, and then the user initiates a coordinated interaction with the scatterplot. First, the two components are added to the coordinator. Next, the Scatterplot receives an interaction from the user. It then informs the coordinator of this, and the coordinator fires an event that the GeoMap receives. Note the “busy” nature of the coordinator in the actual firing of events: all coordination events are sent by the coordinator.
A Component-Based Coordinator – CoordinationManager

The component-based coordinator, in contrast to the object-oriented coordinator, coordinates arbitrary types of events between two or more of Java objects. It does so by requesting registration (and deregistration) of objects based on their class information. By contrast with the object-oriented coordinator, the component-based coordinator (the main class for which is the CoordinationManager) works by registering objects with each other. Events are then sent from object to object without the coordinator interceding in any way.

Java, as a programming language, has the advantage (shared with Microsoft’s new language C#, which was derived from Java) over C++ or Visual Basic that there is extensive, and automatic, class metadata available for any object. This enables us to do meta-programming based on class metadata. For example, we can ask any Java object, what class defines you? For any Java class, we can ask what methods, fields, and constructors do you have? In addition, what
interfaces do you implement? These automatic reporting mechanisms make many powerful meta-
programming approaches possible, such as the coordination explained here, programs as web
services, and incorporation of tools in many contexts (Cazzola, Ghoneim et al. 2002).

Using these class metadata queries, whenever an object (in our example, geoMapOne) is
registered to an instance of CoordinationManager, the coordinator examines
geoMapOne’s class, GeoMap, for methods that can add and remove references to interfaces, for
example, SelectionListener. Then, all previously added beans are examined to see if they
implement the interface, in this example, SelectionListener. If any previously added
beans do implement this interface, then Method.invoke() is called using the appropriate
object identity reference and arguments. In this case, the method is
addSelectionListener(SelectionListener l) declared by the class GeoMap,
using geoMapOne and the previously added bean which implements SelectionListener as
arguments. This registration process is illustrated in Figure 2-29, Figure 2-31, and Figure 2-30.

After this is done, any selections created and passed on by geoMapOne will be passed on to the
listening object. Similarly, GeoMap will be queried to see if it implements any interfaces for
which previously added beans have listeners, and geoMapOne will be registered with them if
any are found.
A component-centered view of the registration and event firing process that takes place in the component-based CoordinationManager is shown in Figure 2-30. The same processes of registering two beans and triggering a coordinated event in the scatterplot are shown. Here, upon registration, each bean is queried via introspection about what types of events it is capable of broadcasting, and capable of receiving. Then, if an appropriate match is found, the bean that is sending is matched with one that is receiving, and the receiving bean is registered with the one
that is sending. This is more complicated than the process outlined above, where each bean is simply registered with the coordinator.

There is a payoff for the harder work of registration. It is that when events are sent, they are sent directly from the sending to the receiving bean, with no intercession by the coordinator. In fact, the coordinator could be removed after registration has occurred, and the beans would function just the same. In a typical analysis session, registration of each bean occurs only once, while hundreds or thousands of events may be coordinated between components. It makes sense, then, to have the coordinator do extra work on set-up to save work while the program is running.

Contrasting the registration process for the GvCoordinator and the CoordinationManager

Figure 2-31 shows the sequence of how beans are registered with the component-based CoordinationManager. In this example, a single component, the GeoMap, is added to the coordinator. First, the user adds GeoMap. Next, the coordinator queries the map about what types
of events the map is capable of sending and receiving. The information is gained by examining all
the methods of the map in turn, and looking for those that match the EventSet naming convention
in Sun’s JavaBean standards. This is done for both the types of interfaces that are implemented,
which show the types of events the map may listen for, and the types of events to broadcast,
which are found by examining which types of event listener interfaces may be registered and
deregistered with the map.

Figure 2-32: UML Sequence diagram of a component being added to a coordinator

Figure 2-31: UML Sequence diagram of a component being added to a coordinator

Figure 2-32, by contrast, shows the simple procedure that occurs when a component is
registered with the object-oriented coordinator. The map is added by a user, and the coordinator
adds the map to its list of coordinated events. This may sound like a component operation, taking
place at runtime, rather than an object-oriented design time solution. However, a key fact to
remember about the object-oriented GvCoordinator is that all components that can potentially
be registered must implement an interface (GvCoordinatorClientListener) that the
GvCoordinator depends on. Therefore, the classes are linked at compile time.
The advantage of the component-based approach is shown in the next two sequence diagrams. Figure 2-33 shows the sequence of a selection change being propagated from the GeoMap to the Scatterplot. The user initiates the change, and the scatterplot receives it, because it was previously registered with the GeoMap.

Figure 2-33: UML Sequence Diagram showing event firing during application use

Figure 2-34 shows the more elaborate process that occurs when the same sequence happens using the object-oriented GvCoordinator. The selection change is initiated by the user,
and the GeoMap informs the coordinator that the selection has occurred. The coordinator then rebroadcasts the event to the scatterplot.

The author placed a few restrictions on the component-based coordination process, to improve its functionality. First, and most basically, objects are not registered with themselves. Secondly, interfaces defined in Java packages (grouped classes in a common namespace) that primarily concern within object communication, rather than between object communications, are excluded. Packages, such as `edu.psu.geovista.app.map` or `javax.swing` are groups of related classes. The packages currently excluded for this reason are `java.awt.event` and `javax.swing.event`. If these packages are not excluded, then mouse clicks and window resizing, for example, would be duplicated across objects, which is (usually) undesirable behavior. Thirdly, the CoordinationManager restricts itself to discovering public methods only. If this restriction is removed, then the CoordinationManager violates Java’s security rules for applets (web-delivered applications running inside other programs), preventing the use
of the CoordinationManager in applets, and preventing any GeoVISTA Studio design that includes an instance of CoordinationManager from being made into an applet. With this restriction, CoordinationManager can operate in applets with no security violation.

**The advantages of the Component-Based CoordinationManager over the Object-Oriented GvCoordinator**

The component-based CoordinationManager has five advantages over the object-oriented GvCoordinator, listed here in order of importance.

1. Stable coordinator
2. Stable clients
3. Limited dependency
4. Runtime control
5. Granular control

Each of these five advantages can be explained with the aid of Unified Modeling Language (UML) diagrams. The advantage will be explained, and the relationship to component-based techniques generally will be made clear.

**Stable Coordinator**

The source code of CoordinationManager does not require modification to be extended to a new type of event. In the object-oriented design of GvCoordinator, if a new type of coordination was desired, the source code for the coordinator would need to be changed, and the class recompiled and redistributed. CoordinationManager, by contrast, can work with future types of coordination that have not been thought of yet, and the CoordinationManager can work with them without modification or even re-compilation of
The Coordinator. The CoordinationManager can do this via the automatic introspection mechanisms mentioned earlier. We can call this the advantage of the “stable coordinator”.

The advantage of the “stable coordinator” is that the source code of the component-based CoordinationManager does not need to be modified in order to accommodate a new type of event to be coordinated. This contrasts with the object-oriented GvCoordinator, which had the event types hard-coded into it. Figure 2-35 shows the object-oriented GvCoordinator on the left, and the component-based CoordinationManager on the right. In these class diagrams, all class members are shown. GvCoordinator has eight event types, and eight corresponding listener lists. If the user wanted to add a further event type to this list, she would be unable to. A developer would be able to add an event type, but would have to modify the source code to do so, which is an undesirable procedure to resort to routinely. CoordinationManager has only two member variables, a list of firing beans, and a list of listening ones. Any new types of events are automatically discovered and coordinated, if there are sending and receiving beans for that event type.
Stable Clients

Client objects do not need to be modified if a new type of event is added. In the object-oriented design, if a new type of event were used, clients would not be able to handle them without their source code being modified. Because events are passed directly from client to client using the component-based CoordinationManager, the events themselves can be extended or otherwise modified without changing old clients or the coordinator. Two examples of this: I changed the selection event to include information about multiple selection authors, that is, I extended what a selection means from a single subset, to multiple subsets with attached authors. In addition, I changed the indication event, which is a transient selection of just one observation,
to include the class of the indicated observation. In both cases, neither the old clients nor the coordination manager had to be modified. We can call this the advantage of “stable clients”.

This advantage, that of “stable clients”, is a complement of the stable coordinator, and stems from the same design considerations. Classes that are to be coordinated by the GvCoordinator have a dependency on GvCoordinatorClientListener, which specifies the events to be listened for. Classes that are to be coordinated by CoordinationManager also have dependencies, however, these dependencies are not directly tied to the coordinator. The difference can be seen in Figure 2-36. On the left, all the clients of GvCoordinator, in addition to the coordinator itself, depend on the interface GvCoordinatorClientListener. In order to add a new event type to those being coordinated, this interface must be extended, and the source code of all clients changed. On the right, all three CoordinationManager clients implement SelectionListener, while two implement IndicationListener. Indications are transient selections showing the observations the user is interacting with. If we hypothesize that we would like our components to start coordinating conditioning, which enables filtering by some variable, possibly not one of those being displayed, then those clients would need to have their source code modified to reflect that support. However, and this is the crucial difference, the other coordination clients would remain unaffected. In our current toolset, we have sixteen coordinated components. It would be onerous, and deleterious to program stability, to change all of them to add a single type of coordination to a single client.
Limited dependency

The CoordinationManager will be more robust than the object-oriented coordinator, in other words, it should exhibit fewer bugs even as it is modified to meet new requirements or the classes it is distributed with change. The CoordinationManager has dependencies on only two other classes, other than depending on the core Java libraries. Both of these classes are in the same package (namespace) as the CoordinationManager. The object-oriented design depended on nine custom classes and implemented a custom interface. We can call this the advantage of “limited dependency”.

The advantage of “limited dependency” is important for robustness in the sense of robustness against change, defined at the beginning of this chapter. We need coordination to be
performed in a way that does not break frequently, so the coordinator class should not depend on any unreliable classes. The more classes the coordinator class depends on, the more likely it is that the coordinator will break unexpectedly if these classes change. Figure Figure 2-37 illustrates why the component-based approach is more robust.

Figure 2-37: UML class diagram showing the differences in dependencies between the object-oriented coordinator (left) and the component-based coordinator (right).

**Runtime control**

The CoordinationManager enables end users to modify how components coordinate, if, for example, users are interacting with an application assembled by Studio. The GvCoordinator does not allow the user any such control without access to Studio. We can call this the advantage of “runtime control”.

The advantage of “runtime control” is a critically important one. In fact, this advantage was one of the primary reasons for developing the component-based coordinator in the first place.
In the process of *Studio* development, end users are not expected to use the visual programming environment that *Studio* provides (see Figure 2-38).

However, without the visual programming environment, the user has no way of controlling how the components interact at runtime. For this, they need a user interface of some kind to customize how the interactions between collaborating components. Figure 1 – 1 in Chapter 1 shows the graphical user interface for the component-based *CoordinationManager*. This allows the user to turn listeners on and off in relation to sending beans. This functioning of this user interface is described in a worked example in Chapter 4.

*Granular control*

The *CoordinationManager* allows a finer level of control over how components interoperate. In the *GvCoordinator*, a component connected to the *GvCoordinator* would send and receive all the possible coordinated types of event. The *CoordinationManager*
allows individual components to send or not send any combination of potentially coordinated events. We can call this the advantage of “granular control”.

The graphical user interface above enables the final advantage, that of “granular control”. This consists of the ability to connect and disconnect listeners from each other using the CoordinationManagerGUI. This allows one to have some types of events shared, but not others. The GvCoordinator did not have this facility, because the linkages between components are defined at compile time (object-oriented), not run-time (component-based).

Figure 2-39 illustrates how this can be useful. In this design, there are two instances of GeoMap. The two maps are coordinated in their color selection, but not in their data sources. This allows the analyst to determine whether patterns evident at one scale are evident at another. Here, using a bivariate color scheme (more fully explained in Chapter 3), we can see that the general patterns of counties with a higher relative proportion of whites and of blacks are similar at the state and county scales, but not identical (with the north-central and northwest U.S. in particularly appearing to have a more uniform race distribution at the state level than is apparent at the county level).
Limitations of the Component-based coordinator

There are some limitations of the component-based coordinator that should be acknowledged. Two are discussed below. Firstly, there is a tendency to pass references to objects as fields of events, which creates shared references to the objects. Secondly, components may become “over-coordinated” if components are broadcasting events with similar effects.

The first potential difficulty comes from the fact that the easiest way to include some information in an event is to add a field that provides access to it. For example, our current SelectionEvent functions just this way, having a method getSelection() that returns an array of integers, notated int (Knoernschild 2002). The potential problem is that multiple coordinator clients may acquire a reference to the same array. Arrays in Java are mutable...
objects, thus any of the coordinator clients can change the array, or assign it a null value, potentially breaking the other coordinator clients. A safer approach would be to have the event have a method that returns an `Iterator`, which is an interface that specifies a means of reading the values. Using an `Iterator` would allow for read-only access to the underlying array. The `Iterator` approach would be slower than the unsafe shared array approach, but probably faster than another safe alternative, which would be returning a “defensive copy” of the array (returning a copy of an object, to prevent the original from being modified) whenever the field is accessed.

Figure 2-40 illustrates the problem and solution in a UML class diagram. The diagram on the left shows the structure of the default approach, which leads to shared read-write references. The diagram on the right shows the structure of the solution, using an `Iterator`.

![Figure 2-40: Class diagram showing different patterns for a selection using shared array references (left) and an Iterator (right)](image)

The second potential problem of the current approach is that it relies on each component sending and receiving the appropriate kinds of events. The default behavior is to coordinate in all possible ways that the components are capable. This can lead to an “over-wiring” situation, with more connections than are desired. Events that have similar meanings are especially problematic,
because if a component “subscribes” to multiple events with similar meanings, later events will undo or change the effects of the first to arrive, without the user being aware of it. For example, if a component is listening for arrays of color as well as classes that contain color finding algorithms (more fully explained in Chapter 3), the one may interfere with the other. A solution to this is less obvious than to the first problem. One option may be to have types of coordination assigned to particular types of task, and have the coordination change to fit the user’s needs (Gahegan 2003).

Development methodologies – Free and Open Source Software

Open source software, or “free as in freedom” software, is a powerful approach for developing geospatial demographic data visualization software. Unfortunately, most GIScience software is not open source. The “flagship” software that ESRI, Intergraph, and Smallworld sells cannot be redistributed, or even investigated too deeply. With closed-source software such as this, following the scientific process of trying to discover exactly how an analytical method works and has been applied (by understanding what the software is doing) may even be a criminal offence! It can thus be argued that if a researcher is using closed-source software, she is not really aware of what she is doing, because the details are hidden. Two popular open source programs outside of GIScience are the web server Apache and the operating system Linux. Important open source GIScience software includes the GeoTools (http://www.geotools.org/) package, and the GRASS GIS (http://grass.baylor.edu//index.html).

The software described here is all open source, with the partial exception of the underlying Java platform. It is open source in two senses; firstly, the software is based on other people’s open source efforts. Secondly, all of the software written as part of this project is being released as open source. Figure 2-41 shows the logos of two of the leading lights of free/open
source software, the GNU (GNU’s Not Unix) project of the Free Software Foundation and the Open Source Initiative. The two organizations will be briefly described in historical order in the next two sections.

What is “Free Software”?  

Free software is a powerful idea, but a somewhat contentious and confusing one. When describing a piece of software as free, the distinction needs to be made between “free as in speech” or “free as in beer”. “Free as in speech”- style free software conforms to the ideals of the free software movement, i.e. software that is released under the GPL (General Public License) or equivalent license, and which is based entirely on software that is also free. “Free as in beer”-style free software is merely distributed without charge. Many free software advocates consider it unethical, or at least highly distasteful, to use any non-free software.

The GPL was written by computer scientist Richard Stallman in response to an unpleasant event where he was unexpectedly unable to use some code he had been relying on. James Gosling (the author of Java) gave Stallman some C code that Stallman used in the Emacs editor he was working on. Gosling later sold the code to a commercial operation, which insisted that Stallman stop redistributing the code in question. Stallman re-wrote and replaced the code, then wrote the GPL to ensure that this kind of situation could be prevented in the future.
The GPL has a few essential provisions, namely, that the source code, and all modifications to the source code, must be made publicly available, and no restrictions may be placed on the further redistribution of the code. People sometimes mistakenly think that software under a GPL license cannot be sold. Redhat and IBM corporations are two well-known companies that re-sell GPL code.

The GNU (GNU’s Not Unix) (a recursive definition) project is a collection of free software utilities, organized by the Free Software Foundation (http://www.gnu.org/). GNU software, in addition to the Linux kernel, forms what is popularly known as the Linux operating system. Linux has achieved a high level of visibility as a Free Software alternative to the proprietary Windows operating system. While the operating system per se is not directly relevant to geospatial demographic data visualization software, the philosophy of Free Software, and Open Source, definitely is.

**What is “Open Source Software”?**

“Open source” as a term is much more recent than “free software”. It described the same process of development as “free software”, but it is a less politically loaded term. Open source software is equivalent to free software in terms of code licensing; there is no difference between the two if both use the GPL license. What the Open Source movement does beyond the Free Software movement is to dispense with most of the political baggage that comes with the Free Software movement. The Free Software movement sees software licensing as an political and ethical issue. They hope to make the world a better place by freeing it from the tyranny of proprietary software. Many business and governmental organizations are hesitant to participate in a development process that is as explicitly political, and somewhat anti-commercial, as this. By contrast, the Open Source movement sees open source software as a practical solution. The Free
Software Foundation describes the difference in this manner: “For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution.” For more information about the difference between the two, see \( \text{(http://www.gnu.org/philosophy/free-software-for-freedom.html)} \). The Open Source movement is trying (with some success) to get open source used in large corporations, where a great deal of software development and use happens.

The OSI (Open Source Initiative) defines open source as follows: “Open source promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. To be OSI certified, the software must be distributed under a license that guarantees the right to read, redistribute, modify, and use the software freely.” (OSI 2003)

According to the Open Source Initiative web site \( \text{(http://www.opensource.org/)} \), the appeal of open source is simple: “When programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, and people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing.” (OSI 2003)

**Open Source Licenses**

There are a number of software licenses that are considered “Open Source”. All of them permit redistribution of source and binary files. Where they vary is in what other restrictions are put on the use of licensed code. From the older to newer, some of the most popular Open Source licenses are the BSD, the GPL, and the LGPL. Here, their basic features, and their advantages and disadvantages, will be described.
**BSD**

BSD stands for Berkeley Standard Distribution. This is a variant of the UNIX operating system. The BSD license merely requires that a copyright notice be included in all redistributions, and that a disclaimer of liability be included as well (California 1999). Thus, code licensed under a BSD or equivalent license (such as MIT or Apache) is compatible with being redistributed and sold as part of a commercial, closed-source software package. This is why a BSD variant is the underlying operating system for Macintosh OS X; BSD-licensed software can be re-packaged and sold, and the vendor can add additional restrictions.

The advantage of this kind of license is that almost no restrictions are placed on the use of the software, just acknowledge that you are using it and refrain from suing the authors, and you are fine! Software released under this kind of license can potentially reach the widest possible audience; if the goal in writing the software is to contribute a pure public good, and there are no concerns about the software being used elsewhere, a BSD license is a good choice. However, if you believe that the open source model is a valuable one, releasing the software so freely may be abetting the “enemies” of open source, such as Microsoft. Microsoft uses BSD code in their TCP/IP handling deep inside their MS Windows products (HPCwire 2001); at the same time, Microsoft vigorously campaigns against the use of open source, having even called it a “cancer” that destroys intellectual property (Wilcox 2001).

**GPL**

The GPL stands for General Public License. The license was written by the Free Software Foundation to ensure that modifications to software with this license remain free. This is the license that the Linux operating system (or GNU/Linux) uses, and the one that best
embodies the Open Source ideals. It requires that the source code for any modifications to the software, and to any software linked to software under the GPL, be made public and redistributable.

The advantage to this is that the developer avoids a situation where she has to compete against a proprietary version of her own work. Thus, other developers are much more likely to want to contribute some time to developing the software, because they know their efforts will remain available as part of the software. The disadvantage of using the GPL is that it is not possible to redistribute versions of the software that mix proprietary and GPL code, unless the proprietary code can be made open source, which many corporations will not find appealing.

**LGPL**

The LGPL stands for Lesser (or Library) General Public License. This license is very similar to the GPL, except that it removes the restriction on redistributing software that is linked to the licensed code, as opposed to derived from licensed code. This makes it more suitable for software that is intended to interoperate with other programs, including commercial, closed-source ones. All the code developed in this research is licensed under the LGPL.

*What about Non-profit?*

Software that allows free redistribution for non-profit purposes only is not compatible with the GPL, or LGPL. The reason for this is that GPL and LGPL guarantee that modifications to open source software will remain open source. Disallowing re-distribution for commercial purposes is an orthogonal restriction to this one; in other words, the two ideas violate each other.
Additionally, placing restrictions on for-profit use limits the potential usefulness of the code, and makes free redistribution potentially problematic.

**For academia**

Ideally, all software created and used for research purposes should be Open Source. How can we evaluate alternative claims without being able to examine what the claims are based on? A frequent complaint about commercial GIS is that they are “black boxes” (GRASS 2003), how they perform many spatial operations is a secret. Since academic interest often centers on the details of how operations are performed, the “how” of software can become the “what” of research.

**Why hasn’t Open Source taken over the GIS market?**

The leading open source GIS software is called GRASS. GRASS has a much smaller market share than ESRI’s products. Why is this? One reason (beyond the originally raster nature of GRASS) might be that different parts of GIS functionality do not meet widely felt needs. The most successful Open Source projects attract developers from many locations because they meet very widely felt needs, like a web server (Apache), or an operating system (Linux).

One middle ground that should be promoted is open standards. Standards allow commercial, non-commercial, and Open Source software to interoperate. This gives us some of the advantages of Open Source software, such as actually knowing what our models and software are doing, with the appeal of being able to reach the large markets that commercial products do. If we cannot move entirely to using open source, the least we can do is push commercial software vendors towards developing against standards. The Open GIS Consortium (OGC) is a leading organization promoting open standards in GIScience.
Summary

Our overall goal is to create robust geospatial demographic data visualization software.

Three important sub-goals were identified at the beginning of this chapter:

- to provide a concise, but complete, guide to UML grounded in design of geospatial demographic data visualization software.

- to explain a novel and useful approach to helping software objects work together (the component-based coordinator introduced in Chapter 1), in sufficient detail such that other researchers could re-implement it in other languages, and to explain the advantages of this component over previous coordination techniques.

- to explain why an Open Source software development model is the best one for development of geospatial demographic data visualization software, especially in an academic research environment.

These goals were each addressed in this chapter.

UML addresses the first need, providing a means of communicating about software design decisions. An introduction to UML, with examples of every major UML diagram type, is presented. Each UML diagram example is drawn from an actual piece of geospatial demographic data visualization software designed as part of this research.

Each diagram type is then used to illustrate the solution to the second sub-goal, to develop software design strategies that allow software components to be connected or disconnected, in flexible ways, at run-time. A strategy for performing such coordination, based on automatic querying of components (introspection), and calling of methods discovered at runtime (reflective invocation) is presented. The benefits of this strategy over a compile-time, object-oriented approach are detailed. These benefits include a more stable coordinator, more stable coordination clients, reduced overall dependencies, and increased end user control at run-time, including finer-grained control over how the components are coordinated.

The coordinator, and the components that are coordinated, are being released under an Open Source license. The final section in this chapter addresses what Open Source software is,
what its history is, and why software created as part of academic research in particular should be released under an Open Source software license. Standards, which promote interoperability, are a viable halfway house where Open Source and proprietary, closed software can meet.

Future directions for this component of the research will focus on expanding the roles that coordination can take, and continuing to investigate new software development paradigms that may aid in the creation of robust geospatial demographic data visualization software. Coordination of both visual and numerical states between components is a key unsolved problem in geographic visualization, and one that our coordinator strategy can help to address. Coordination may also be able to partially address the problem of saving a collection of heterogeneous components, along with their state information, to storage for later retrieval. Future development will also enable more flexible types of coordination. A possible means of doing this is using aspect-oriented programming (Kiselev 2003), such as with AspectJ (http://www.eclipse.org/aspectj/) to map types of coordination to types of analysis tasks.
References Cited


Chapter 3

Geovisualization, InfoVis and EDA: Transforming geographic space and data space into user space.

One of the central problems in geospatial demographic data visualization is that geospatial demographic data spaces are often too large and complex to be directly represented easily. To provide useful visual access to highly multivariate data, two essential ingredients are required: first, the data must be presented to the user in a way that allows sensible visual access to some of the structure (or lack of structure) inherent in the data, and second, the analyst must be provided with a variety of perspectives on the data. Three research domains that have spent considerable effort trying to solve these problems are geographic visualization (geovisualization), information visualization (InfoVis), and exploratory data analysis (EDA).

Effective geospatial demographic data visualization requires that the principles and techniques of these disciplines be applied. Doing so will address the overall challenge, to provide interactive control over multivariate, geo-referenced data space and thus support flexible exploration and knowledge construction. More specifically, goals in this research related to each research discipline are:

- **Geovisualization** – to create appropriate interactive geographic components for multivariate demographic data visualization.

- **InfoVis** – to leverage graph forms that allow the user to interactively browse over highly multivariate data spaces.

- **EDA** – to provide interactive tools that allow the user to see multivariate data spaces.

Meeting each of these goals yields general novel visualization strategies, and in turn, a usable coordinated suite of tools that is the result of applying those strategies.
These chapter goals match with the overall dissertation goals in the following ways. The geovisualization chapter goal contributes to meeting Goal 2.1, of providing flexible data-to-color mappings. The InfoVis goal helps to meet Goal 2.2, of reducing the complexity of highly multivariate data spaces. The EDA goal is essentially identical to Goal 2.3, of directly representing highly multivariate data spaces.

The strategies used to meet the goals and the implemented tools are presented in three sections. The first section gives some background on the three academic disciplines, and explains how the approaches presented here combine and extend them. The second section introduces a set of direct geovisualization techniques. Direct geovisualization techniques are geospatial representation methods for which all entities represented in a data set, and all relevant aspects of these entities, are present in their own visual (or other sensory) representation. Direct geovisualization becomes impossible for data sets that are too large, either in number of observations, or in number of variables. The second section presents direct visualization techniques that draw on work from geovisualization, information visualization, and EDA. In the third section, interactive approaches that combine computation and visualization for data sets that are too large for direct visualization are presented. Application of these approaches results in a novel node-and-edge graph based technique that draws on geovisualization and Infovis.

The direct visualization approaches that address the geovisualization goal are two novel geovisualization methods and associated software components. These methods apply to highly multivariate data spaces, and are implemented as concrete tools that can be used by any interested data analysts. The first method focuses on creating color spaces of arbitrary complexity, the instantiation for which is a color picker that flexibly maps variables to colors. The second method focuses on using bivariate mapping techniques, including use of complementary colors, the instantiation for which is a bivariate mapping component that can be used either as a stand-alone tool, or as an element in a multi-representation matrix. The multi-representation matrix is a tool
that can accept different types of bivariate visualization methods, and display them, using each element’s position in the overall matrix to control the variable shown.

The color techniques are also applied in the context of creating a direct geovisualization approach that draws representation concepts from InfoVis. Here, I propose a novel exploratory visualization method and associated tool that extends existing InfoVis methods. The technique is a space-filling visualization representation that enables novel bivariate arrangements of data. The technique is implemented in a proof-of-concept pixel-based space-filling visualization tool, which can be plugged into multi-representation matrices, as the bivariate map can.

I also propose a direct geovisualization idea that extends methods originally introduced in EDA, based on the concept of a parallel plot. The concept is a directly manipulable parallel coordinate plot with easy control over classification and coloring of polylines. The novel useful tool that reflects this idea is a parallel coordinate plot that is well suited for geospatial demographic data visualization, and has the novel feature of allowing external ordering of axes to automatically place more related axes near each other, as well as easy, integrated control of color and classification.

The third section, on approaches to geographic representation of data sets that are too large for direct geovisualization, introduces a novel technique for interactive browsing of large data spaces. This technique relies on combining computation and visualization, calculating the relationships between observations in high-dimensional space, and then presenting them as an interactive node-and-link graph. The instantiated tool is called Linkgraph. Linkgraph also incorporates the color and classification techniques presented in the next section, allowing the analyst to compare the distribution of any given variable in the data set with the set of variables chosen for node-and-link graph analysis.
Background on Geographic Visualization, Information Visualization and Exploratory Data Analysis

The problem of how to visually represent multiple (more than three) variables has been a focus of attention in many domains, including EDA (Fisherkeller, Friedman et al. 1974; Inselberg 2002), InfoVis (Keim and Kriegel 1994; Keim 2000), and geospatial demographic data analysis (Haug, MacEachren et al. 1997; Kwan 2000). Approaches to this problem also forms one of the important advantages of the novel techniques developed as part of this research. This section begins by considering two general issues in development of multivariate visual analysis tools, data dimensionality and representation abstractness. Then geovisualization, InfoVis, and EDA are compared briefly. Each is then considered in more detail.

Data dimensionality

We can put the problem of multivariate data visualization in context by starting with attention to univariate visualization, and moving up, as in Figure 3-1 below. This table draws from the taxonomies of Buja et al. (Buja, Cook et al. 1996), Keim and Kriegel (Keim and Kriegel 1996), and Wegenkittl et al. (Wegenkittl, Loffelmann et al. 1997).
The current research has contributions to make at each group of number of dimensions. For the one-dimensional case, a system of color anchors is described which gives the user interactive control over color ramps. This functionality is encapsulated in a component with a graphical user interface and integrated classification facilities, the Visual Classifier. The Visual Classifier can be “dropped in” to a variety of visualization components. It is integrated with all the tools described in this chapter. For the two-dimensional case, the color anchor system is extended to work with bivariate components, including a bivariate map component. The bivariate mapping component also can be used inside multi-form matrices, which can be used in the four to

<table>
<thead>
<tr>
<th>Number of Dimensions</th>
<th>Numeric representation</th>
<th>Geographic representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Histogram, Box Plot</td>
<td>Choropleth map, map with single visual variable</td>
</tr>
<tr>
<td>2</td>
<td>Scatterplot, Bag Plot</td>
<td>Map with two visual variables</td>
</tr>
<tr>
<td>3</td>
<td>Scatterplot with additional visual variable</td>
<td>Glyphs in maps, Coordinated representations, Matrices</td>
</tr>
<tr>
<td>4 to 7</td>
<td>Traces, Glyphs, Matrixes, PCP, Radviz</td>
<td>Glyphs in maps, Coordinated representations, Matrices</td>
</tr>
<tr>
<td>More than 7</td>
<td>Dimension reduction techniques to fewer variables, then as above</td>
<td>Dimension reduction techniques to fewer variables, then as above</td>
</tr>
</tbody>
</table>

Potential direct geovisualization techniques

Figure 3-1: Table showing kinds of representations and dimensions for geographic and numeric representation methods, and potential direct geovisualization methods. Methods which can potentially represent each datum of a total geospatial data set of interest, are identified as potential direct geovisualization techniques.
seven dimensions cases. Additionally, for this range of number of dimensions, a parallel coordinate plot is developed which can have its axes ordered to aid in identification of interesting patterns.

For representations that show seven or less variables, direct geovisualization (representation of all observations, and all variables of interest) is normally possible. Seven is chosen here as a stopping point for the number of variables that can be usefully visualized at the same time, based on the classic work of Miller (Miller 1956), who identified the magic number 7, plus or minus 2, as the number of separate concepts that can be held in the mind at one time.

Different display techniques may be able to usefully display more variables than others, and the usefulness of different displays may vary with the size of the dataset.

“Concrete” and “Abstract” representations in geovisualization

In addition to a distinction between methods that support direct geovisualization and those that require statistical summary, a distinction can also be made based on degree of abstraction. For example, exploratory data analysis approaches can be separated from ones derived from information visualization techniques based on their degree of abstraction. If there is a measurable relationship between the representation and the quantity it represents, we can call this a “concrete” data representation, if there is not, we can call the representation an “abstract” data representation. A thought experiment for testing the abstraction of a representation is to imagine taking a ruler or a light meter and measuring data points, and writing down their values, in effect reconstructing the data table that the representation was derived from. If this reconstruction can be accomplished, the representation is concrete, if not, the representation is abstract. Geovisualization, InfoVis, and EDA approaches all use both abstract and concrete
representation methods. As a generalization, InfoVis approaches are more abstract; EDA approaches are more concrete, while geovisualization approaches span the range.

Abstractness is always a matter of degree. There must be some relationship between the data and the representation, if only an indirect or abstract one, for the representation to be useful in understanding the phenomena the data represents. The most abstract might be those that give categorical information only, in other words, those that give information about which observations are similar to each other, but not the order of the observations, for example, tree maps from InfoVis and Mosaic plots from EDA both fall into this category. Most concretely, we can think of a dot plot where the data are mapped as dots along an axis.

Close examination of the dot plot should enable us to create a table similar to the one the dot plot was based on. Even the dot plot, however, introduces a greater level of uncertainty when compared to the original tabular presentation of the data. A given dot might be obscuring another data point. We often also lose some information about precision in the graphical view. As a point of medium abstraction, consider a classed choropleth map. The data that the colors are based on can be reconstructed to the level of the category for each observation (Peterson 1979). At a further point of abstraction, other representations, such as “pixel-based” ones (Keim and Kriegel 1994; Hinneburg, Keim et al. 1999; Keim 2000) give us information about the order, on some relative scale, but not about the absolute value.

There is a relationship between direct geovisualization and the degree of abstraction in the representation. If a data set cannot be represented directly, then some degree of abstraction is necessary to represent the totality of the data.
Comparing geovisualization, InfoVis, and EDA

Geovisualization is an academic discipline that “provide(s) theory, methods, and tools for visual exploration, analysis, synthesis, and presentation of geospatial data” (MacEachren and Kraak 2001). It is a key area of research for realizing the benefits from the enormous investment in digital access and information stores of geospatial data our society has made over the last ten years. To understand what geovisualization is and how it contributes to the development of geospatial demographic data visualization software components, along with how it differs from cognate fields like exploratory data analysis (EDA) and information visualization (InfoVis), first the origins of geovisualization and its relations to other fields will be described.

Geovisualization has its roots in cartography, EDA is a research focus in statistics, and InfoVis has its roots in computer science and information science. The three approaches have much in common. They all use computer graphics to pre-process and to display data, and they all try to exploit human visual pattern recognition abilities to reveal unknown patterns in data. However, they differ in both their subjects of analysis and their preferred visual methods. We can represent these disciplines by assigning an iconic software tool to represent each one. These tools are not meant to be restricted to development or use in one community, but to represent the types of data and the types of visual analysis most common in each community.

A representative application for geovisualization is an interactive map (MacEachren and Kraak 1997), with the ability to control how the geographic data are symbolized on the map (e.g., a choropleth map that supports dynamic re-classification). The interactive map symbolizes the tools and the methods at the heart of geovisualization. The tool is a graphic one that has a “world in miniature”. The method is that the user can adjust and control her view on that world, to bring knowledge into focus. For InfoVis, the representative application is a hyperbolic browser (Lamping and Rao 1996), a method of displaying and manipulating arbitrarily large hierarchies of
information in a tree structure (discussed in more detail below and illustrated in Figure 3-3). This tool presents potentially large and complex data sets in a user-centered manner. The box-and-whisker plot (Tukey 1977), also discussed in more detail below and illustrated in Figure 3-4), which summarizes statistical data, represents EDA. Easily constructed, compact, and informative, the box-and-whisker plot, like other EDA techniques, has inspired both InfoVis and geovisualization.

**Geovisualization**

Geovisualization, geographic visualization, or cartographic visualization as it used to be known, began in the late 1980s, but its academic roots go back further. It can be seen as a natural outgrowth of the application of computing technology, the scientific visualization approach of mapping data to visual attributes, the exploratory spirit of EDA applied to cartography, and of Bertin’s graphic information processing approach of the 1960s (MacEachren 2001).

Cartographers have been interested in using computers to produce maps for more than 40 years. Initially, use of mechanical devices to produce maps went under the name of “automation” (Tobler 1959; Sherman 1961). Cartographers were interested in using electronic calculators to help them with the many calculations that are necessary to produce a map; automated printing of maps came later. Some of the early research foci were interpolation using Delaunay triangulation (Boots 1986), text placement (Imhof 1962; Robinson and Sale 1969; Imhof 1975; Wood 2000), terrain shading (Yoeli 1967; Lewis 1992), and demographic mapping (Tobler 1959). An early example of automated printing is shown below (Figure 3-2). This figure shows a map created (by Bertin in 1967) by fitting a typewriter with special symbols, and then using punch cards to read in the data.
Showing one of Bertin’s early automated maps is appropriate for the introduction of geovisualization in another way as well. Ideas that Bertin pioneered in 1967, but were not introduced into the United States until his work was translated in 1981 (Bertin 1981), lie at the heart of geovisualization. These include using visual displays of geographic information to generate hypotheses, and to filter and explore the data spatially (MacEachren and Ganter. 1990;
MacEachren 1998; MacEachren 2001). EDA and InfoVis researchers point to Bertin as a pioneer for their fields as well (Card, Mackinlay et al. 1999; MacEachren 2001; Friendly 2002).

One route that these ideas took built upon the EDA tradition of using graphical methods to explore numerical data started by Tukey (DiBiase 1990; Macdougall 1992; MacEachren and Monmonier 1992) (described in more detail below). Traditional cartography had been about communicating knowns; the revolution of geovisualization was that it is using dynamic geospatial representation to find unknowns (MacEachren 1994; MacEachren, Wachowicz et al. 1999). Some examples of how this has been put into practice in interactive, exploratory environments include (Dykes 1998; Andrienko and Andrienko 1999b; Harrower, MacEachren et al. 2000).

Early sources on geovisualization include two books (Visualization in Modern Cartography (MacEachren 1994) and Visualization in Geographic Information Systems (Hearnshaw and Unwin 1994)). Since then, there have been special journal issues devoted to geovisualization, including such journals as Computers and Geosciences (1997), the International Journal of Geographic Information Science (1999), and Cartography and Geographic Information (2001). There has also been a EURESCO conference devoted to geovisualization (Methods to Define Geovisualisation Contents for Users Needs in Albufeira, Spain, 2002) with a successor planned for Greece in 2004. A good source for the current state of geovisualization is the International Cartographic Association (ICA) Commission on Visualization and Virtual Environments (MacEachren and Kraak 2002).

Geovisualization ideas have also been extended into sound (Krygier 1994) and touch (Krygier 1994; Griffin 2000), as well as into software objects for inclusion in a geographic information system (Frank and Egenhofer 1992). Dykes’s cdv environment includes cartograms, or maps whose areas are not geographic but instead tied to other factors like population. Cartograms remain an active area of geovisualization research (Keim, North et al. 2003) and an effective visualization technique for population data (Tobler 1986; Dorling 1993; Upton 1994;
Dorling 1995; Edelsbrunner and Waupotitsch 1997; Kocmound 1997). The inverse of cartograms, taking data spaces and turning them into map forms, is called spatialization (Fabrikant 2000; Skupin and Fabrikant 2003). Cartograms and spatialization are an example of changing data space to user’s space, which is a common strategy for information visualization.

**InfoVis**

Information visualization (InfoVis) approaches can inform the design of geospatial demographic data visualization in a number of ways. InfoVis researchers have created an astonishing variety of data representations, only a subset of which are of interest in the current research. Among the important methods from InfoVis for design of geospatial demographic data visualization are how to represent node-and-link graphs, how to present abstract data spaces, and creation of novel user interaction techniques.

As noted above, a software tool that symbolizes the subject and methods of InfoVis is the hyperbolic browser. It takes an abstract structure, like the connections between web pages, and transforms it into a usable, graphical representation. Figure 3-3 shows an example of a hyperbolic browser displaying an organization chart, represented as links and nodes.
InfoVis researchers claim Bertin as a forefather, because they have adopted the philosophy evident in *Semiology of Graphics* (Bertin 1983), which shows ways of mapping data onto various aspects of information displays. An example of this approach in a geographic setting is denoting qualitatively different categories of cities with differently shaped symbols.

This abstract mapping of data to visual representations is what separates InfoVis researchers and practitioners from the Visualization in Scientific Computing community; where scientific

---

1 Geovisualization researchers claim him as a forefather as much or more for ideas presented in *Graphics and Graphic Information Processing*
computing usually takes a natural physical representation as its base (e.g. depicting air flow in the atmosphere), InfoVis applies visual processing to abstract information (e.g. depicting the relationships between different textual documents) (Munzner 1997).

As in the hyperbolic browser example above, methods for visualization of abstract spaces like graph visualization are important to InfoVis (Huang, Eades et al. 1998; Bridgeman, Garg et al. 1999; Bridgeman, Di Battista et al. 1999; Stasko, Catrambone et al. 2000; Vismara, Di Battista et al. 2000); for a survey of graph visualization see (Herman, Melancon et al. 2000). Similarly, other abstract spaces that can be usefully represented using Information Visualization techniques include the structure of the internet (Rohrer and Swing 1997; Mukherjea 1999; Abel, Gros et al. 2000; Pohl and Purgathofer 2000), of internet browsing behavior (Osawa, Asai et al. 1999), of organizational structures (Zhang 1998; Brandes, Kenis et al. 1999), and of large datasets (Keim, Kreigel et al. 1999; Keim 2000). Techniques from these applications can be leveraged in geospatial demographic data visualization as well.

Besides a focus on representing abstract relationships, InfoVis directs considerable attention to user interface issues related to productive use of these representations. These issues include improving interactivity (Roth, Chuah et al. 1997), in both 2D and 3D representations (Chuah, Roth et al. 1999; Robertson, Card et al. 1999; Risden, Czerwinski et al. 2000), using video in InfoVis (Christel and Martin 1998), and investigating the potential of virtual environments (Chen, Czerwinski et al. 2000).

In common with EDA, InfoVis researchers are interested in novel ways of linking different data representations (Eick 2000; North and Shneiderman 2000), while the use of maps (Hewagamage and Hirakawa 2000) connects InfoVis to geovisualization. An important emerging research theme for all three is usability (Hartson 1998; Chen and Yu 2000; Chen and Czerwinski 2000; Graham, Kennedy et al. 2000).
An online information visualization bibliography is available at (Munzner 1997; UIRG 2001). The leading North American meeting on InfoVis, the IEEE Information Visualization Symposium, has their conference programs and email records at [http://www.infovis.org/](http://www.infovis.org/). For further detail on InfoVis, an excellent one-volume survey of InfoVis is provided by *Readings in Information Visualization: Using Vision to Think* (Card, Mackinlay et al. 1999).

**EDA**

EDA developed as a graphical approach to exploratory statistical analysis. A visual approach to data analysis that symbolizes the tools and philosophy of EDA is the box-and-whisker plot (Figure 3-4). In this plot, “M” represents the median, or mid-point, of the data, with 50% of the observations above it and 50% below. “Q1” and “Q3” mark the first and third quartiles, with 25% of the observations below the first quartile, and 25% above. The lines extending outwards from the quartiles (the whiskers) show the limits beyond which observations are considered outliers, represented by 3/2 of the difference between the first and third quartiles (interquartile range). Beyond this, each observation is represented as a dot.

---

![Box and Whisker Plot](image)

Figure 3-4: Box and Whisker Plot. Image by (Weisstein 2003)
The seminal work in exploratory data analysis in statistics is John Tukey’s book, *Exploratory Data Analysis* (Tukey 1977). The principles advocated by Tukey include an exploratory approach to learning what characteristics exist in a data set and using graphic display to harness our natural abilities to detect patterns. The patterns that are of interest are often statistical ones, like mean, standard deviation, and trends.

One commentator defined EDA as an approach to data analysis which employs a variety of mostly graphical techniques to:

1. maximize insight into a data set;
2. uncover underlying structure;
3. extract important variables;
4. detect outliers and anomalies;
5. test underlying assumptions;
6. develop parsimonious models; and
7. determine optimal factor settings (Filliben 2001).

A key to the EDA approach to software, and one that has been adopted by geovisualization as well, are the facilities of linking and brushing. An early advocate of these techniques is Cleveland (Cleveland 1987) (Cleveland 1994), as well as Carr (Carr and Glass 1989), Buja (Buja, McDonald et al. 1991), and (Shneiderman 1994). A more recent survey is found at (Buja, Cook et al. 1996). The leading journal for EDA is the *Journal of Computational and Graphical Statistics*, although EDA has become mainstream enough in statistics that the Journal of the American Statistical Association contains many examples of exploratory graphical analyses, for an example, see (Spitzner, Marron et al. 2003).

EDA, InfoVis, and geovisualization techniques and ideas can all help to answer research questions, detailed at the beginning of this chapter, on the best methods for visualization of georeferenced enumerated data. These techniques, along with implementations of these techniques, are described next.
Novel Direct Geovisualization Techniques

As part of the current research, a mapping component was developed and implemented to meet the specific goal of supporting highly multivariate demographic data visualization. Two of the research goals that the development helps to meet are: giving the analyst flexible control over color selection, specifically allowing different color ramps to be applied over different parts of a data distribution; and packaging these color selections for application in a wide variety of components. To address the more general goal of support for highly multivariate analysis, a number of novel techniques and their implementations are presented here. These techniques are presented from most abstract to least abstract in their presentation of numerical data (using the definition of abstraction defined above). All of these techniques are capable of directly presenting all observations in a data set.

The most abstract, from the perspective of numerical data, is a system of color assignment called “color anchors”. These are then applied to a map component. The map component has been designed to support univariate and bivariate representation and for the bivariate maps to work within matrices, where they can be paired with other bivariate visualization components. One of these other bivariate components that can be presented in a matrix is based on a space-filling visualization technique that has been developed and implemented here. The third technique, which represents numerical data most directly (concretely), consists of extensions to the idea of a set of parallel axes with observations represented as lines between them. This technique is implemented as a parallel coordinate plot component, which also leverages the color tools developed here.
Flexible User Control of Color Schemes

Figure 3-5 shows the color anchoring system that was developed as part of the current research. I introduce the concept of multiple color anchors as a way to control key points in a color sequence while allowing the advantages of color ramping to still be applied for generating any arbitrary number of colors intermediate to the anchors. The anchors help the user construct complex paths through *color space* that appropriately emphasize parts of the data distribution being represented. A color space defines color in terms of standard numerical attributes; this allows different devices to produce the same colors, and allows operations, such as interpolation, to be performed. Color spaces recognized by the International Organization for Standardization (ISO) include RGB (Red Green Blue), CMYK (Cyan, Magenta, Yellow, and Black), and CIElab. Each of these has different purposes. RGB is the most commonly used space in computer graphics. CMYK is used in printing. CIElab is designed to be a perceptual color space, in which equal distances in the color space are perceived as equal differences by the human eye. RGB space is used in most of the examples provided here. Calculating CIElab color spaces is difficult, however, work is ongoing to incorporate perceptual color spaces with the tools mentioned here. Note that the anchoring system described here is compatible with any color space in which values can be interpolated.

The use of color anchors helps the user construct color schemes that better fulfill Trumbo’s “Separation” principle (one of four principles discussed in more detail below). This principle is that “Important differences in the levels of a statistical variable should be represented..."
by colors clearly perceived as different.” (Trumbo 1981). If we accept that the most important
differences are those defined by the user, then allowing the user to create more separation, using
color anchors, in whatever part of the distribution the user desires should be seen as an advance in
applying this principle to color schemes. Brewer (Brewer 2001) discussed the need for custom
color schemes that highlight particular breaks in the data and that have imbalance with more steps
of one hue than steps of another hue in a diverging scheme (and illustrated the advantage of doing
so in an award-winning atlas – (Pickle, Mungiole et al. 1996)). The bivariate counterpart to this
argument will be explained below in reference to bivariate maps.

Color anchors are used inside a component called the ColorRampPicker. The
ColorRampPicker performs a linear interpolation in a color space (such as RGB) between all
anchored points. The endpoints are always anchored. If no additional anchors are added, the ramp
is a linear interpolation between these two endpoints. Additional anchors introduce additional
points between which a linear interpolation is performed. This enables the analyst to visually
highlight any part of the data distribution of interest, while retaining the ability to see
differentiation in other parts of the distribution.

Figure 3-6 illustrates how the anchors operate in two-dimensional color space. There are
two anchors in addition to the required ones at the endpoints. These two anchors produce greater
color contrast at the beginning and end of the color ramp. This results in two “steep” color
gradients, with a large “shallow” gradient section in the middle.
The component called the “Visual Classifier” encapsulates the functionality of taking data, classifying it into groups, and assigning symbolization to it. The latter matches data to attributes of what Bertin termed “visual variables” (Bertin 1981). Thus far, the only type of symbolization that is supported is color fill and texture overlay for polygons, but in the future it is planned that many more types of symbolization, such as size of glyphs, shape, orientation, textures, and dynamic behaviors, will be supported. The most important advantage of the visual classifier, as currently implemented, is that it is a stand-alone component that can also be embedded in other components. In the current toolset, the Visual Classifier appears in the Map, Bivariate Map, Parallel Coordinate Plot, Linkgraph, and Scatterplot components. When additional functionality is added, for example, support for the ColorBrewer (Brewer and Harrower 2002) color choosing methods, that functionality is automatically added to all components that make use of the Visual Classifier.
The class structure which helps make the Visual Classifier addable to a variety of geospatial demographic data visualization software components can be better visualized in a UML class diagram (Figure 3-7). In this diagram, the position of the classes is significant; parent components are higher up than the components they contain. A parallel coordinate plot and a map both have a Visual Classifier, which in turn has a Classifier Picker and a ColorRampPicker. The ColorRampPicker is in turn composed of a set of RampSwatch components. Note that the connections from components lower down to components higher up, for example from a Visual Classifier to a map, or from a RampSwatch to a ColorRampPicker, are via abstract listener interfaces. This means that the contained components do not know anything about the components containing them; this is what enables the components to be easily applied in many different contexts, without specialization for each case.
The user interface to this component is shown in Figure 3-8. It has been designed to be compact, while providing the essential functions of mapping data to symbolization choices. As noted above, the visual classifier is made up of two smaller components, a classifier picker (top), and a color picker (bottom). The parts of the classifier picker, from left to right, are a slider bar that chooses the number of classes to apply, an editable text box that displays the number of...
classes to apply and that provides an alternative means to specify that number, a selectable drop-
down list for which classification method to use, and a drop down list of which variable to
classify on. The color picker has a number of color patches (in this case, four) that can show
themselves a horizontal or vertical orientation, and an anchor icon to indicate the fixity of the
color when in the anchored state.

Figure 3-8: Visual Classifier

Univariate Map Component

The map component for visualizing a single variable essentially marries a visual classifier
to a graphics rendering area that can display multiple layers of polygons, along with a toolbar for
controlling the mode of interaction and the spatial extent being currently shown in the map. The
map toolbar allows the user to toggle between four modes, and take one action. The four modes
are selection, zoom in, zoom out, and pan. The action is to zoom to full extent, which is indicated
by a house (home extent). Figure 3-9 shows the map component with the color anchor choices
described above. The variable being displayed is the population density of each county in 1990.
The distribution is broken up into twenty quantiles, and the two highest categories of counties and
two lowest categories of counties are visually highlighted by the use of color anchors. This
highlights a number of features that would be difficult to notice with any single linear ramp. The
strip of counties in the middle of the country that have particularly low populations, from the
Dakotas to west Texas stands out. At the same time, major metropolitan areas such as Los
Angeles, San Francisco, and Seattle, stand out, along with their suburbs. This color anchor
method is one of the methods that enables data-to-color mappings that fit Goal 2.1, identified in the introduction, of finding the means to give the user interactive, flexible control over such data-to-color mappings.
Bivariate Color Schemes

In the 1970s, a series of bivariate color maps were produced at the U.S. Bureau of the Census (see (Meyer, Broome et al. 1975) for details of how they were produced). These maps were criticized for being difficult to interpret. In response, Olson (Olson 1981) conducted a series of experiments to determine how well the bivariate coloring schemes used on the Census maps
could be interpreted by map users. She found that map users were able to interpret the maps, and that the map users had generally positive attitudes towards the maps.

Olsen also found that users did not find the color schemes used to be intuitive (i.e., given the colors for a scheme, few users could identify the logical arrangement required to build the appropriate legend). Once the schemes were explained, however, the participants could understand their logic. This result suggested that the concept of a bivariate map was understandable, but that there could be more effective and understandable color schemes. This in turn suggests that giving the user interactive control over color scheme creation may aid in improving scheme comprehensibility. Additional support for giving interactive control of bivariate color schemes to the user was provided by Rheingens. She found experimentally that bivariate maps with dynamic control of color were more effective than their static counterparts (Rheingans 1992). She introduced just such an interactive system, called Calico. Calico differs from the work described here in that it focused on constructing both paths and surfaces through 3D color space. The implementation described here may also be easier for many users because it does not require understanding of color spaces nor interaction with a 3D view.

Tumbo, in earlier work on bivariate maps, focused on deriving the qualities that a bivariate color scheme ought to have, and proposed four principles for color schemes, two that apply to univariate and bivariate color-to-data mappings, and two additional principles that apply to bivariate mappings only (Trumbo 1981). The first two principles are those of order and separation. Order means: “if the levels of a statistical variable are ordered, then the colors chosen to represent them should be seen as preserving that order”. Separation means: “Important differences in the levels of a statistical variable should be represented by colors clearly perceived as different.” Interestingly, this principle of separation does not explicitly exclude n-class maps, as long as the color scheme represents significant differences in the data clearly.
The two principles Trumbo offers specifically for bivariate schemes he calls “rows and columns” and “diagonal”. “Rows and columns” means that the two variables in the bivariate scheme should not interfere with each other. “Diagonal” means that if one of the goals of the map is to show positive association, there should be three classes evident in the map, a class showing correlated observations, those with observations relatively higher in one variable, and a class showing observations with relatively higher values in the other variable. Trumbo critiques the Census schemes on the basis of both of these principles.

**Bivariate Map Component**

The map component, combined with the bivariate color scheme tool described above, can function as a bivariate visualization device. Especially when complementary colors are used, the result can help the analyst find spatial relationships between two variables at once (Eyton 1984; Olson and Brewer 1997). Figure 3-10 illustrates an instance of this. The two variables shown in the figure are the proportion of people in counties in the U.S. who identified themselves as Asian or Pacific Islander, shown in green, and the proportions of persons who identified themselves as white, shown in purple, both from the 1990 U.S. Census. Both variables have been divided into three quantiles, with an equal number of counties in each group. Through the bivariate classification, we then have nine combinations.

Because the color scheme uses complementary colors, the counties that are low in proportion of Asians and low in proportion of whites come out as light gray, while the mid-mid and high-high combinations are represented as darker grays. This can lead to problems of axis scaling and be potentially misleading if the relationship is not both linear with a slope of one, and with a Y-intercept of zero (Carstensen 1986). For such data, grays may not appear even though two variables are closely related. These problems are, however, mitigated by using quantile
classifications. The color hue combinations shown here are similar to ones found to be preferred by analysts in a study of diverging color schemes for univariate maps (Brewer, MacEachren et al. 1997). Those places that had relatively higher proportions of Asians as opposed to the proportions of whites are colored green. Those counties that have a relatively higher proportion of whites are colored in purple.

The bivariate patterns revealed are a mixture of the expected and the unexpected. The south, where much of the nation’s black population lives, has relatively the lowest proportions of both whites and Asians. There are a higher proportion of Asians, relatively, along the west coast and in the New York – Washington corridor. Whites form a relatively higher proportion of the population in Appalachia, in the “heartland” belt from Pennsylvania to Indiana, and in the northern Great Plains. Perhaps more surprising is the high-high proportion of whites and Asians in the middle of Idaho and of Michigan, as well as the relatively higher proportion of Asians in Florida.

It is important to note that although the quantile classifications used here create equal numbers of observations in each of category for individual variables, the bivariate scheme does not apply the constraint that there be equal numbers in all bivariate categories. In fact, one of the more useful features of a bivariate scheme such as this is that if the two variables are largely positively correlated, the overall map will be gray, if we use quantile classifications. If the two variables are negatively correlated, then the map will consist mostly of the bivariate colors (purple and green in the illustrations shown). A video example of a number of variables being examined in the bivariate map component can be seen at (http://www.geovista.psu.edu/members/hardisty/videos/geomap.avi).
The bivariate color combinations described above can be combined with color anchors to create potentially helpful bivariate color schemes, such as the cross maps suggested by Monmonier (1992) in which a binary classification is used for each variable resulting in a four-
class bivariate map. Figure 3-11 illustrates how these combinations can be made with color anchors, and how they may be useful in the context of geospatial demographic data visualization. The flexibility provided by color anchors allows particular data ranges to be highlighted. In this map, the two variables being mapped are the proportion of the 1990 population in each U.S. county that is self-identified as black and the proportion self-identified as white. The classification scheme divides both variables into five equal quantiles. The color anchors help to highlight three types of counties. Firstly, counties that are in the lower fifth for both proportion of whites and proportion of blacks are shown in white. There are not many of these. Second, the counties with low proportions of blacks and not-low proportions of whites are pink. Thirdly, the counties with low proportions of whites and not-low proportions of blacks are green. This map shows black-white segregation at a national scale.
As noted above, the functionality shown in Figure 3-11 is essentially that of “cross maps” (Monmonier 1992), which have also been implemented in the Descartes system (Andrienko and Andrienko 1999a) and the HealthVisB system (MacEachren, Boscoe et al. 1998). Cross maps are essentially bivariate maps with direct manipulation of the joint class break point. Monmonier presented this mapping method in a non-interactive map-movie prototype, Descartes providing an extension into fully interactive classification. The color anchor system presented here goes beyond what the Descartes system is able to do, because it is a more flexible tool. Descartes makes it easier to manipulate the specific class break points while the color anchor system makes it easier to manipulate specific color characteristics. In the Descartes interpretation of cross maps,
the user can pick a number of classes for each variable, a classification method, and a single color scheme. There is no way for the user to alter the color scheme. Similarly, the earlier HealthVisB prototype system has a fixed color scheme, but it allows the user to quickly step through 5% breaks in each variable.

Allowing for flexible mixing of color schemes allows for variations like those in Figure 3-12. The variable and classification scheme are the same as in Figure 3-11, the only thing that has been changed is the bivariate color scheme. Figure 3-12 highlights the same basic bivariate pattern as Figure 3-11, but the addition of gray scales allows for the visualization of some patterns not shown in the first figure. For example, in Pennsylvania, the fact that there are higher relative proportions of whites than of blacks, except in the southeast and southwest is shown.
Figure 3-13 illustrates further the flexibility of bivariate color mapping using color anchors. Here, the color ramping is done in HSB (Hue, saturation, and brightness) space instead of in RGB space. It can be argued that this scheme offers superior performance in relation to Trumbo’s “rows and columns” principle for bivariate color schemes. The rows, which represent
different quantiles of the proportion of the population identifying as black, stand out distinctly.

The columns, consisting of differences in the proportion of American Indians, do not stand out as well, but are still perceptible. The “diagonal” principle is not well served, but this may actually be desirable in cases where there is no positive correlation between the variables. The preceding methods of interactive control over bivariate data-to-color mappings meet the goal, identified in the introduction, of finding such methods.
Figure 3-13: A bivariate color scheme in Hue, Saturation, and Brightness color space, emphasizing counties with different quantiles of proportions of blacks, while showing some of the variation in other counties in the proportion of American Indians. Those places with a high relative proportion of American Indians are shown in shades blue, with middling proportions of American Indians in shades of green, and with low proportions of American Indians in shades of brown. Places with a higher relative proportion of blacks are shown in darker colors.
Matrices

The map component was designed to fit into matrices of components. The full names for these components are Multiform, Bivariate Matrix (Figure 3-14, left side) and Multiform, Bivariate Small Multiples plot (Figure 3-14, right side) (MacEachren, Dai et al. 2003). These may be referred to as (name, name) matrix and (name, name …) small multiples, respectively, where the (name, name) are the names of the plots within the overall matrix, for example, Map and Scatterplot Matrix and Map, Spacefill, and Scatterplot Small Multiple. The overall matrix was designed to be able to accept any kind of bivariate plot to reside in each cell that could fulfill a specific “contract.” This contract defines what a plot residing within a matrix cell should be able to do. The maps in the matrices follow a bivariate color scheme as described above, taking one variable from their location on the X axis, and the other from their location on the Y axis. When matched with scatterplots, the scatterplots serve as legends for the maps (or other display forms).
The manner that components are added to the matrix is through a Java interface that specifies the responsibilities of the components that reside in the matrix, which are referred to as elements of the matrix. This allows the matrix to remain open to allowing different types of components to be added to it. Figure 3-15 illustrates the relationship between the matrix and the elements that go inside it. There are currently three matrix element types, a map, a scatterplot, and a space-filling visualization component. Through the matrix element interface, it will be possible to add more bivariate map types in the future.
Space-Filling Visualization

Space-filling visualizations are transformations of user space that try to maximize the space used by each observation. Important categories of this type of visualization include Tree-maps, Sunburst visualizations, and pixel-based displays. Each is introduced below and an example with geographic data is presented.

Space-filling Displays

This section presents a tool inspired by a technique called, variously, pixel charts, or spiral charts, or space-filling visualizations. The application domain is population data for enumerated units such as counties. Some previous presentations of these space-filling displays are appropriately called pixel charts, because one observation was mapped to one pixel. Thus, data
sets can be visualized that have as many data points as we have pixels available. On the other hand, similar techniques may also help when there are fewer values to examine.

Figure 3-16 shows a detailed version of the space-filling component implemented as part of the current research. Along the top, there is a visual classifier (introduced earlier in Figure 3-8), which allows the user to classify data and map the classes to colors. Since the Visual Classifier is an independent component, as the design and functionality of the Visual Classifier improves, by adding support for better color choosing and for more visual variables, like shape and size of symbols, this will be automatically reflected in the tool as a whole. Below the Visual Classifier, there are pick lists for choosing three ways data are mapped to visual appearance, the “color by” variable, the “order by” variable, and the type of fill order. Unlike the case where we map one observation (or more than one observation) to one pixel, mapping one observation to more than one pixel makes it possible for display units to have other shapes, or to have superimposed shapes as shown here. A video demonstration of the user interface for this spacefill component can be seen at

(\url{http://www.geovista.psu.edu/members/hardisty/videos/spacefill_1.avi}).
Figure 3-16 uses the default, *scan-line*, order. Using this method, the most conceptually straightforward, observations are ar ranged by rows from low (bottom) to high (top). The lowest value is at the lower left of the bottom row, the 7th lowest (with this layout) is at the right of that row, the 8th lowest is at the left of the second row from the bottom, and so on. In Figure 3-16, the observations are U.S. states that have been ordered by their population in 1990. Therefore, the state with the lowest population (Wyoming) is first, followed by other low population states. The same order is followed along the top, so the last state in the upper right, California, is the one with the highest population. The squares representing states are colored according to their land areas.
There is a mild correlation (0.52) between land area and population, as shown in Figure 3-16. Most of the whitest states are in the lower part of the graph.

The observations may be laid out in many different configurations. These generally fall under the heading of space-filling curves. Space-filling curves are most commonly used to map two-dimensions into a single dimension. Here, they are used in the reverse capacity, to map a single dimension onto two. Figure 3-17 shows the three space-filling curve types that have been implemented as part of the current research, and are accessible from the graphical user interface. The first method is the one described in detail above, scan-line order. The second is a Hilbert curve, which recursively fills larger and larger squares. In this context, the Hilbert curve has the disadvantage of leaving blank up to one-quarter of the display space. Figure 3-17 is nearly such a case. Thirdly, we have a spiral arrangement, which starts with the lowest observations in the center, and spirals out. This is the layout favored by many of those doing pixel-based displays of this kind discussed here (Keim 2000).
Using space-filling visualizations in geospatial demographic data visualization

The primary advantage of space-filling techniques is that they prevent over-plotting, or the obscuring of some data points by others. A scatterplot with many thousands of observations will inevitably obscure many of them. The novel features of this software are shown in the UML class diagram for the spacefill component, Figure 3-18. These include the ability to include the display in various kinds of matrices, as well as the integration of the common color and classification tool, the Visual Classifier.
Matrix Arrangement

The first advantage of our pixel-oriented component is that it can be arranged in a matrix, varying the “color by”, and “arrange by” variables simultaneously. Researchers in space-filling visualizations have previously tried matrix representation of pixel-oriented displays (Keim, Ming et al. 2002), but did not vary both the variable to be used for arrangement of observations and the variable used for coloring, in different cells of the matrix. A spacefill and scatterplot matrix component, which does vary the arrangement variable and the coloring variable by cell, can be seen in Figure 3-19. The “arrange by” variable is determined by the individual visualization component’s column position, and the “color by” variable is determined by the component’s row position. In the scatterplot, the position of all the observations is determined by their row and column. The data are U.S. Census statistics by county. Seven variables are shown, in order, population density, proportion of males, proportion of blacks, proportion of American Indians /
Eskimos, proportion of Asians/Pacific Islanders, proportion of other race, and proportion of Hispanic ethnicity. People, in the U.S. Census categorization, can be Hispanic and of any race. 3100 counties are shown (Puerto Rico is not included here). Combinations for each variable are shown by county, so excluding the diagonals, there are \(42 \times 3100 = 130,200\) distinct data points being shown simultaneously, not an easy feat for most visualization devices.

Figure 3-19: Matrix of scatterplot and spacefill visualization components, showing race/ethnic structure in U.S. counties in 1990. Spacefill-scatterplot comparisons discussed in the text are highlighted.

Some relationships in Figure 3-19 are evident in both a given scatterplot and in its corresponding spacefill visualization, some relationships are evident in the scatterplot that do not show up in the spacefill visualization, and some relationships are evident in the spacefill visualization but not in the scatterplot. An example of a strong positive relationship that is evident
in both, is the relationship between the proportion of the population that identified as “Hispanic” and the proportion identified as “Other”, which is the pair of graphs in the lower right, outlined in light blue. The scatterplot shows a strong positive relationship, as evidenced by the points trending in an upward slope. We can see the same relationship in the spacefill visualization in three bands, with most light gray counties concentrated towards the bottom of the graph, and darker purple at the top. The bottom counties are those with a low proportion identified as “Other”, and the light gray counties are those with a low proportion identified as Hispanic, so the positive relationship is shown by the low to high, light to dark pattern of the bands.

The scatterplots can show some relationships that the spacefill visualizations do not. For example, in the scatterplot depicting the proportion of whites and the proportion of blacks, outlined in dark blue, there is a sharp line running from the top left to the bottom right. This shows that there is a limit to how high the combined proportions can be (the total percent of black + white cannot exceed 100%), and that there are counties composed of only whites and only blacks at almost all combinations that add up to one hundred percent. This relationship cannot be directly seen in the spacefill visualization. However, there is a hint of it in the thick purple band along the bottom, and a much thinner white band at the top. The thick band means that those places that have a high proportion of blacks almost all have a low proportion of whites, while the thinner light band means that those places with a low proportion of blacks are predominantly, but not exclusively, those with a high proportion of whites.

Conversely, there are relationships that the spacefill visualizations show that the scatterplots do not. This is especially the case with data sets that have thousands of observations, like this one, because of the over-plotting in the scatterplots. Take, for example, the spacefill visualizations that show the relationship between the proportion of Asians compared with whites and blacks. Both of the spacefill visualizations for these relationships show white bands at the top and at the bottom. This means that among those places with a very high or very low proportion of
blacks or whites, there tend to be few Asians. This relationship is difficult to detect in the corresponding scatterplots. Interactions with this same dataset can be seen in a video demonstration at (http://www.geovista.psu.edu/members/hardisty/videos/scatterspace.avi).

Figure 3-20 shows a matrix of only space-filling visualization components. This matrix of space-filling visualization components shows some more details of the racial structure of the U.S. Having the entire matrix filled up with spacefill visualization components is helpful because the ordering and the coloring are being done on different bases; corresponding spacefill visualizations across the diagonal will not necessarily be identical. For example, the pair of spacefill visualizations that show the relationship between the proportions of whites and of blacks both show an inverse relationship, but in the spacefill ordered by proportion of blacks, the counties at the top of the spacefill visualization are entirely light gray. This shows that those counties with a high proportion of blacks never have a middle or high proportion of whites. The corresponding light gray band in the spacefill visualization ordered by the proportion of whites is a bit more variegated, showing that those places that have a high proportion of whites sometimes have a middling high proportion of blacks. This is probably an artifact of the way the data are being classified, because the “middle” category for proportions of blacks (5% to 9% of the population) extends to a much lower proportion than the middle category for whites (91% to 98% of the population), so it is numerically impossible to for places that have a relatively high proportion of blacks to have a middling high proportion of whites. Such artifacts are worth finding, if the visualization devices are to be well understood. The presentation of space-filling visualization components in the matrix partly meets the goal, identified in chapter 1, of combining pixel-based visualization techniques with other visualization forms.
Sharing the Matrix with other Components

Secondly, our pixel-oriented component can be paired with an arbitrary number of other bivariate components, such as a scatterplot or map, in a matrix. Both kinds of representation will use the same set of variables as already shown in Figure 3-19. This enables the user to have immediate access to a familiar representation such as a scatterplot, which may be of particular importance to users unfamiliar with pixel-oriented displays. Alternatively, using the map in conjunction with the pixel-oriented component, users can rapidly compare geographic patterns with patterns shown in the pixel-oriented display.
For example, Figure 3-21 shows U.S. Census data at the county level, with maps above the diagonal and space-filling components below. Both the maps and the spacefill visualizations are bivariate, and they are given the variables they are to visualize based on their position in the overall matrix, but they use different pairs of visual indicators to show the data values. The maps use a complementary bivariate color scheme as described in relation to Figure 3-10 above. Thus, in the maps, the amount of purple is determined by the column, and the amount of green is determined by the row. In the spacefill visualizations, the visual characteristics are color and order, not color and color. The order variable is taken from the column position of the spacefill visualization, and the color variable is taken from the row.

The power of combining the maps with the spacefill visualizations in one matrix component is fully realized only when the interactivity provided is taken advantage of. See (http://www.geovista.psu.edu/members/hardisty/videos/spacefill_map_matrix.avi) for an example. The analyst may select a certain geographic place and see where these observations occur in statistical space, or conversely, select some data range in a spacefill, and see what the geographic places are that correspond to that range.
Adding symbols

Thirdly, we have (experimentally) developed the ability to place other symbols on top of the expanded pixel displays (see Figure 3-22). Here, the outlines of the states have been inserted, in order to aid in their identification. This use of shapes in combination with a space-filling display would not be useful at the county level; it would be a rare geographer who could identify all the counties of the U.S. based on their shapes (!). However, using different shapes, sizes, and colors of glyphs to carry information holds considerable potential. For example, Figure 3-22 has translucent gray dots over three states, simulating representing uncertainty about those values.
Coordinated exploration

Fourthly, our pixel-oriented component is linked to many other information visualization and geographic visualization components via a coordinator (discussed in Chapter 2) to allow colors, selections, and other user driven display changes to be automatically propagated and shared amongst them. Use of this dynamic linking among tools is covered in more detail in Chapter 4. This integration is the other method that meets the goal, identified in Chapter 1, of combining pixel-based visualization techniques with other visualization forms.

Figure 3-22: Space-filling graph showing U.S. states, contrasting the gender/race relationship between black and American Indians. Translucent gray dots have been added to simulate uncertainly information.
Applying EDA: Parvis

Recall that EDA is primarily a method for direct visual representation of numerical data, where the quantities being represented have direct, measurable counterparts in the displays created. A prototypical EDA tool is the parallel coordinate plot, or PCP (Inselberg 1985). A PCP presents a great deal of detail about the data, and at the same time provides a holistic view of the relationships among variables and among data points. Parallel coordinate plots have been used to explore massive data sets, such as analyzing gene expression (Zhang, Tang et al. 2003). The method is potentially quite interactive (Wegman 1990; Edsall 1997; Siirtola 2000; Andrienko and Andrienko 2001a). The PCP as implemented here is novel in its combination of two aspects: it has an integrated visual classifier, and it allows the reordering of axes based on any external service. The close coordination with specifically geographic representations makes this a direct geovisualization technique.

What is a PCP

A PCP can be thought of as a scatterplot with many parallel axes (instead of two orthogonal axes, one for X and one for Y). Observations are depicted with multi-part lines between the axes (instead of dots on the scatterplot). A worked example is seen below in Figure 3-23. The figure depicts two entities, for each of which there are three variables. One entity is marked in blue, the other in green. Note that in a PCP, crossing lines show negative correlation, (i.e. as one quantity goes up, the other goes down) and parallel lines show positive correlation.
Previous presentations of PCPs

Inselberg was the first to publish on the subject of parallel coordinate plots. He has continued examining the statistical properties of these plots and extending their functionality (Inselberg 1985; Inselberg, Chomut et al. 1987; Inselberg and Dimsdale 1994; Inselberg 1998; Inselberg 1999; Inselberg 2002). (Cook, Majure et al. 1996) added links to and from a map to their parallel coordinate plot in the XGobi toolkit. (Dykes 1998) used a PCP in conjunction with a map to represent spatial autocorrelation measures. Some possibilities for using PCPs as a means of combining Knowledge Discovery in Databases (KDD) with geovisualization were explored (MacEachren, Wachowicz et al. 1999), including assigning colors to the strings in a PCP. (Siirtola 2000) pointed out some ways that direct manipulation may be applied to a PCP interface. The Descartes toolkit (Andrienko and Andrienko 2001b; Andrienko and Andrienko 2001a) has implemented a linked PCP as one of its visual analysis elements. The Descartes PCP is especially noteworthy in the variety of axis scaling techniques that are explored, including normalization of axes by median, mean, and standard deviation values, and scaling to straighten the line of any selected observation or pair of observations. This last method provides the user with an easy

<table>
<thead>
<tr>
<th></th>
<th>Variable 1</th>
<th>Variable 2</th>
<th>Variable 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Data 2</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3-23: Data table and associated parallel coordinate plot
means of focusing in on ranges of interest. The selection of which variables to display in Descartes PCP is also well integrated with other multivariate display options.

The PCPs described above are powerful visual analytic devices, but they can still be improved upon. Research challenges to be addressed include: How to support conditioning (described below) in the context of a PCP and how to give the user more flexible control over the color space illustrated. Both of these challenges are addressed by the development of the PCP described here. Conditioning support, which allows filtering the visual analysis based on ranges of related variables (Carr, Wallin et al. 2000), is an easy conceptual match with the parallel coordinate plot. Coloring the lines on a PCP using flexible color picking should improve the amount of information a PCP is able to convey. The development leveraged existing software components, including the Parvis PCP, the Visual Classifier described in Figure 3-8, and a component for managing conditioning.

**Parvis PCP**

The PCP shown in Figure 3-24 is called Parvis (Hauser, Ledermann et al. 2002; Ledermann 2003). The initial instantiation was authored by Flo Lenderman. Some of its desirable properties include: direct manipulation of axes for axes reordering, axis scaling, and axis translation. Architecturally, it was created as a stand-alone component in Java, a JavaBean. This makes it easy to work with in a component programming environment like Studio. It is also an attractive base on which to build because it has an open source license that permits others to extend the component, as long as their changes are made public. A substantial limitation in Parvis as implemented by Ledermann was that it was designed as a stand-alone visualization component that was not able to send and receive information like selection, indication, and visual aspects of data presentation.
An interesting aspect of the original Parvis PCP, which has been maintained in the adapted one, is direct manipulation of axes in novel ways. Previous PCPs have allowed the user to reorder axes using the mouse and a drag-and-drop interaction metaphor (Gahegan, Takatsuka et al. 2000; Masters and Edsall 2000). Parvis extends direct manipulation beyond this, to allow the user to change the scale of the axis in two important ways. First, the user can translate the axis scale, dragging up to change the data range higher, and down to change the data range lower. Second, the user can change the axis zoom by dragging in a similar way. Dragging up zooms out, and dragging down zooms in. These direct manipulations are more intuitive than bringing up a dialog box to some users, and allow the analyst to change important numerical properties of the visualization display in visual manner. These direct manipulations are arguably equivalent to using the mouse to pan and zoom in a map display, which would now be considered essential functionality for an interactive map visualization component.

The discussion below highlights some of the extensions I have made as part of this dissertation research. The extensions that were made include adding more general data import routines (Parvis initially used only one, custom, data format), and adding color and classifications.
A novel PCP

Figure 3-25 shows the user interface as extended for the current research. There are three novel aspects of the resulting PCP, one internal and two external. The internal aspect is the integrated Visual Classifier, the two external ones are support for axis reordering and for conditioning, as driven from components outside the PCP.

Figure 3-24: Parvis PCP user interface
An addition to the internals of Parvis is the integration of the Visual Classifier. This gives Parvis the ability to use arbitrarily complex color spaces, and have the data automatically mapped to those spaces. For example, in Figure 3-26, we can see the result of expanding the number of classes and adding anchors. Both displays are of the same variables in the same order, and both displays are colored according to the percentage of males in each county, the only difference being the color ramping used. The bottom PCP has a more varied color ramp emphasizing the high end of the distribution. It shows us that, while the counties with the highest percentages of males (gold) are fairly spread out in relation to the proportion of blacks in those counties, those counties with medium-high proportions of males (in light purple) are almost all low in proportion of blacks. The integration of the Visual Classifier partly meets the goal of developing a PCP that

Figure 3-25: Extended Parvis PCP
supports integrated classification and symbolization. Use of the Visual Classifier within Parvis can be seen in an example video at

(http://www.geovista.psu.edu/members/hardisty/videos/pcp_1.avi).

Note that the integration of the Visual Classifier has the side effect of allowing Parvis to easily respond to the types of changes that the Visual Classifier affords from outside the component. For example, a map component with a Visual Classifier can also be driving the classification and color choices used in Parvis.

Figure 3-26: Two PCP displays showing U.S. population data at the county level. The bottom PCP display is colored using a flexible color picker.

Parvis has also been extended to accommodate two important classes of input from other components, namely, conditioning and external axis reordering. Conditioning is a “filtering out”
of observations based on the range of some variable, or of some geographic places (Carr, Wallin et al. 2000). Conditioning is often done based on a different variable than is being displayed, which can provide insight into the relative distribution of the remaining observations and can allow the analyst to remove known sources of variation in the data in order to explore the unknown.

Figure 3-27 illustrates how conditioning can aid in the analysis of a data set. Two PCP displays are shown, which are identical except that the top one shows all counties in the U.S., and the bottom one is conditioned to show only those counties that have at least 5% of the population of that county identify as American Indian. By eliminating those other observations, we can see the relationships between gender ratio and the proportion of American Indian in U.S. counties more easily. For example, the counties with at least five percent American Indians are concentrated in the middle range for proportion of males, and at the low end for population density.
The second advantage that is external to the modified Parvis PCP, is that it supports axis ordering from external sources. The axis ordering facilities provided by the Subspace Linkgraph component (described later, in relation to Figure 3-37) can be used here to automatically order the axes in a manner that will enable the analyst to more easily discover relationships. The original Parvis supported manual reordering; the advance described here is to allow statistically based reordering. A video of these two components working in concert can be seen at (http://www.geovista.psu.edu/members/hardisty/videos/subspace_pcp.avi). Other means of grouping variables can also be used, from principle components analysis (PCA), to conditional entropy (Basharin 1959; MacEachren, Dai et al. 2003). The support for axis ordering and

Figure 3-27: PCP with support for conditioning. The bottom PCP shows only those strings with at least 5% of the population identifying as Native American. This reveals some patterns that these places have, such as relatively low proportions of blacks and a small range of proportions of males.
conditioning partly meets Goal 2.3.2 of developing a PCP to support a common classification and symbolization system.

Geovisualization of Highly Dimensional Data Spaces

The overarching strategy adopted in this research as a means of making highly multivariate data spaces accessible to the analyst is to rearrange the data space presented to the user, to help enable the user to see multivariate patterns. Such strategies are called for particularly when the data spaces that the analyst wishes to consider are so highly multivariate that the direct geovisualization approaches described previous in this chapter cannot be used. A kind of visualization technique that can aid the analyst by such rearrangement is using node-and-link graphs. This technique can show the relationship among multiple variables at once, between different sets of outliers, and between outliers and sets of variables. This technique may also be linked with geographic maps in order to bring out the spatial patterns in what the user sees.

The current research develops a strategy for using node-and-link approaches with a direct manipulation interface, implements that strategy, and presents a proof-of-concept application. The sections below introduce graphs, graph visualization, and a novel interactive graph visualization tool called Touchgraph (developed elsewhere) and its extensions developed as part of this dissertation.

Node-and-Edge Graph Visualization of Large Data Spaces

Cartographers have long faced the problem of how to create maps that have many data values available for each observation location; for example, when analyzing causes and consequences of poverty at the national scale, we might have hundreds of relevant variables
available for each county. This problem is especially daunting when the relationships between the
different variables are not known clearly, precisely the situation for which visualization methods
are claimed to be useful. One strategy for approaching this situation is to imagine that the data
have been projected into a multi-dimensional data cube (a hypercube), with one dimension for
each variable. Then, one can find numerical relationships, i.e. distances, among the points in the
high-dimensional cube. These points can be considered as nodes, and the distances between them
as edges, in an abstract node-and-edge graph.

What Graphs Are

Node-and-edge graphs are fundamental structures in computer science and mathematics. Graphs consist of a set of nodes and a set of edges, where each edge consists of the connection between two nodes. There are conferences and journals devoted to the study of graphs and to the sub-field of visually representing graphs in ways comprehensible to humans (Tamassia, Dibattista et al. 1988; Herman, Melancon et al. 2000).

Graph structuring of data is so useful that it underlies some of the most powerful representations of both raster and vector views of spatially referenced data. Efficient raster data formats such as quad-trees are graphs; the “tree” in quad-tree refers to a graph with no cycles (cycles being defined as a set of links connecting a node with itself, with no repeated links). The Tiger data format (as well as the DIME format which preceded it) also is made up of graphs, the “Ti” in Tiger stands for topologically integrated; the topology is represented using graphs.

The fact that graph structures are used in these disparate contexts does not stem from any underlying unity of these data structures, but rather from the fact that graphs are flexible enough to represent a wide range of data structures, and that graphs are useful enough to make it worthwhile to fit these data structures to the graph format.
Graphs and geospatial demographic data visualization

When considering how to represent a demographic data set, for example, U.S. county level data from the 2000 census, graphs have the potential to help. If we are considering all counties (3,233) and all of the long form variables (over 1000), we have a large data space. Representing this as a table does not allow us to view all the data at once. Conventional graphical representations like scatterplots can only represent two or three dimensions at a time. Using a graph representation allows us to see the relationship between arbitrary numbers of variables at once.

As noted above, we can view any numerical data set as a graph by taking each observation as a point in a hypercube defined by the relevant variables. Each variable can be thought of as a dimension. If we map a single variable, the data space can be thought of as a one-dimensional line. Adding another variable makes the data space a two-dimensional plane. Three variables define a cube of data space. Additional variables define hypercubes with increasing numbers of dimensions, which are perhaps difficult to conceptualize, but not to work with computationally.

As a preparatory step to turning a data set into a hypercube, each variable is transformed into a uniform range, for example, from 0 to 1. If this were not done, then variables with larger ranges, such as total population would overwhelm the rest. Then, the distance is found between each pair of points in that hypercube. Next, we define each observation as a node, and edges as the relationship between two nodes. The combination of all the observations and their relationships can then be treated as a graph.

There are varieties of different graphs that are potentially of interest. From those with the most edges to the least, the kinds of graph that have been implemented or extended as part of this research are the Complete Geometric Graph (G), the Delaunay Graph (DG), the Gabriel Graph
The geometric graph is the set of all edges. The minimum spanning tree is the set of edges that have the shortest total length, connecting all nodes without any cycles. The Delaunay and Gabriel graphs lie between these extremes (Cai, Sochats et al. 1999).

Delaunay graphs, based on a set of edges, can be defined as the collection of edges “satisfying an ‘empty circle’ property: for each edge we can find a circle containing the edge's endpoints but not containing any other points” (Eppstein 2003). Delaunay graphs, and the inverse structure, Voronoi polygons, have many applications in geography, including as a basis for point pattern analyses (Boots 1986), for forming triangulated irregular networks (TINs), and as a basis for cellular automata models (O'Sullivan 2001). Most of these cases have been for two dimensions, for reasons specified below.

**Delaunay and Gabriel graphs**

The Delaunay and Gabriel graphs are of interest in the current context because they provide more information than the minimum spanning tree. The MST does not display much local structure, which the Delaunay and Gabriel graphs may capture. While Delaunay and Gabriel graphs are conceptually useful, and in the case of Delaunay graphs, are fundamental to cartography and GIS, their use for analysis in multi-dimensional data spaces has been limited. The use of Delaunay and Gabriel graphs in high-dimensional spaces has proved impractical for two reasons: computational complexity and graph density. The Delaunay graph derived from any complete Geometric Graph G contains the Gabriel Graph. The Gabriel graph derived from a graph G can be defined as follows: two points p and q from graph G are connected by an edge of the Gabriel graph if and only if the hypersphere defined by points p and q does not contain any other points (Gabriel and Sokal 1969). In a planar graph, there are a number of methods for
efficiently calculating Gabriel graphs, however, the algorithms in higher dimensions are not nearly as efficient. One of the most computationally inexpensive ways of calculating these graphs in higher dimensions is to project the hypercube onto a hyperboloid with one more dimension, and then find the convex hull (Okabe, Boots et al. 2000). For an illustration of this projection procedure, see Figure 3-28. In this figure, $P, P_1, P_2,$ and $P_3$, are points on a two dimensional surface, and $P^*, P_1^*, P_2^*$, and $P_3^*$, are the same points that have been lifted onto a three-dimensional spheroid. The convex hull is computed in the spheroid, and this allows us to determine that point $P^*$, and thus $P$, are outside the circle defined by $P_1, P_2,$ and $P_3$. This “lifting” transformation eases the computational burden, because it considerably reduces the number of pairwise points that need to be considered.

The algorithms using this approach are of computational complexity $O(n^{\lceil t/2 \rceil + 1})$, where $n$ is the number of observations, and $t$ is the number of dimensions. Therefore, if we wish to calculate the Delaunay triangulation for U.S. Counties in a data set containing 30 variables, it would take
about $4.3 \times 10^{55}$ convex hull point-in-polygon operations. Considering that a state of the art dedicated Java computer with four-way processing can achieve 25,087 operations per second (in 2001), this would take about $5.5 \times 10^{43}$ years! This is several orders of magnitude longer than the universe has been in existence, and too slow for a responsive graphical user interface.

From the point of view of visualizing the graph, even if it were possible to calculate the Gabriel graph efficiently, visualizing the results would be difficult, because in higher dimensions, the Delaunay and Gabriel graphs have too many edges, approaching the Geometric case. Figure 3-29 shows how the number of edges can increase in just a few dimensions. In Figure 3-29, there are fourteen nodes (shown as points) embedded in a three dimensional space (x, y, and z). The fourteen points are divided into two collinear groups. The two lines are at right angles to each other. Thus, all seven points along the x dimension should be connected to all of the points along the y dimension in a Gabriel graph, yielding 49 edges in addition to the edges between the collinear points, or 61 in all.
This is a somewhat contrived example, but the problem it illustrates becomes even more severe in higher dimensions. For example, if we calculate a Gabriel graph for 50 observations (the States of the United States) based on simple counts of six racial variables (number of whites, blacks, etc.), we end up with 831 links. The resulting visual depiction, at least as rendered here, suffers from the difficulty of over-plotting (Figure 3-30). Examining this graph, we can conclude that New York, Texas, and California are different, but it is hard to extract much more information. Since the Delaunay graph is a superset of the Gabriel graph, the Delaunay graph would have even more edges, and thus suffer more from the problem of over-plotting.
How to Construct a Minimum Spanning Tree

One strategy for addressing the over-connection limitations of Gabriel and Delaunay graphs, as discussed above, is to apply a minimum spanning tree approach. The set of edges that
connect all points, with the minimum total edge length, without creating any cycles, is the minimum spanning tree. Figure 3-31 illustrates how the algorithm for finding a minimum spanning tree works. The first step is to connect the two nodes that are closest together. This means connecting nodes 1 and 2, with edge 1. Next, we connect each unconnected node to the node closest to it, using the shortest edge first, then the next shortest, and so on, until all nodes are connected to at least one node (Hartigan 1975). In the example shown, this means connecting nodes 4 and 5 using edge 2, then nodes 1 and 3 using edge 3, and finally nodes 4 and 6 using edge 4. Now, we connect any unconnected node groups, again using the shortest available edge in each case. In the example shown, there is only one connection to be made in this step, which connects nodes 1 and 4, using edge 5. Note that there is one less edge than there are nodes. This is always the case using minimum spanning trees.

Figure 3-31: Example of a Minimum Spanning Tree (MST) being constructed
The above procedure always finds the shortest set of edges that will connect all nodes, assuming there is a unique solution. It is, however, not the most computationally efficient procedure, taking $O(n^t)$ time, where $n$ is the number of entities, and $t$ is the number of dimensions, the $t$ term entering because distances are recalculated between points. It also is expensive in terms of the number of objects to be created, because it depends on first constructing the complete Geometric Graph. For example, if one wishes to find the minimum spanning tree for U.S. Counties, there will be approximately 3,300 nodes, and about 5,400,000 undirected edges in the Geometric Graph. There are a number of promising approaches to more efficiently calculate minimum spanning trees. The best known is to project the data into a space with an additional dimension, and then calculate the convex hull in that space, from which the Delaunay triangulation can be derived, as shown in Figure 3-28. This gives a much-reduced set of edges from which to search for finding the minimum spanning tree. The problem with this approach is that it still requires a very large lattice to record all the information in. Some other promising approaches include genetic algorithms (Zhou and Gen 2003), an agent, or “ant-based” approach (Shyu, Yin et al. 2003), and parallel algorithms (Chong, Han et al. 2003). Currently, the tool implemented does not apply any optimization approaches, and thus is limited to data sets with 800 observations. In the future, a hyper-cell binning approach will be explored to make the tool scale up to much larger data sets.

Using Minimum Spanning Trees to understand highly multivariate data spaces

The overall goal for the development and implementation of the minimum spanning tree technique within the GeoViz toolkit is to create a visual tool that allows the user to browse over large information spaces, looking at how geographic places are related in an abstract data space that is not amenable to direct representation. Since these are necessarily summarizing
representations, focusing in on the observations that are nearest each other in the data space makes sense. Additionally, MSTs are one of the oldest and most commonly used and studied graph structures, making them a good default choice.

Minimum spanning trees have been used as a clustering technique for summarizing data spaces for some time (Hartigan 1975). Murtagh (Murtagh 1985) maintains that two strengths of MSTs are that they form partitions between classes that are optimal with reference to the connectivity criterion used, and that small changes in input produce small changes in output, i.e. the results are stable. Minimum spanning trees were used by a group at Korean National Open University, working with Di Cook, as part of an interactive data reduction process (Kim, Kwon et al. 2000). They found that MSTs were more effective than dendrograms at helping to identify influential observations in the clustering process. Their implementation gives interactive control in the sense that the user had buttons to push (Figure 3-32), which then affect the display parameters.

Figure 3-32: Graphical User Interfaces for interactive MST tool (Kim, Kwon et al. 2000)

While the MST tool developed by the group at Korean National Open University is interesting, the Linkgraph tool goes beyond it in interactivity. The Linkgraph tool developed for
this research provides for the direct manipulation of data points to browse data space. The application developed by Korean National Open University has an indirect interface, where the user can push a button and see the display updated. This is less direct than letting the user grab and rearrange the nodes for themselves. Additionally, the Linkgraph allows for coloring of nodes based on any variables in the data set, perhaps a variable that was not used to form the minimum spanning tree. This allows the user to make observations about the relationship between a large set of variables and a single one.

An example application of a minimum spanning tree applied to a simple geographic problem is provided in Figure 3-33 below. Here, the minimum spanning tree is found for the X and Y location of the centroid of each state. We can see some of the strengths and weaknesses of the minimum spanning for showing relationships between observations. The minimum spanning tree is effective in highlighting groups of proximate places (nodes near one another in the graph are actually near one another in XY space). Washington is actually close to Oregon, Michigan to Wisconsin, Maine to Vermont, and so on. However, distant nodes in the graph are not necessarily distant geographically. For example, California’s node is shown as being far away from Oregon’s, even though those two states are adjacent in reality. Thus, when this method shows a relationship existing (signified by a small distance between nodes in multivariate space), it does in fact exist; however, places distant in the graph are not necessarily distant in multivariate space.
As noted above, a primary goal of this dissertation is to develop strategies for visually-enabled multivariate geospatial data analysis, implement those strategies, and demonstrate their effectiveness through proof-of-concept application to real data about which real questions are being asked. Here, I introduce two variants of a MST and illustrate their potential applications independently and when dynamically linked (through the coordinator described in chapter 2) to a map (and other components).

Figure 3-33: Minimum spanning tree of U.S. states based on longitude and latitude of state centroids

**Linkgraph**

As noted above, a primary goal of this dissertation is to develop strategies for visually-enabled multivariate geospatial data analysis, implement those strategies, and demonstrate their effectiveness through proof-of-concept application to real data about which real questions are being asked. Here, I introduce two variants of a MST and illustrate their potential applications independently and when dynamically linked (through the coordinator described in chapter 2) to a map (and other components).
The graph representations shown in Figure 3-30 and Figure 3-33 were created with a tool developed by extending an Open Source graph component – Touchgraph (Shapiro 2003), and linking it to other tools (e.g. MST calculation, color selection) as an application. Touchgraph is an interactive component for viewing graphs. Touchgraph has excellent facilities for interactive exploration of graphs: one can drag nodes around, and other connected nodes will trail along after them, and the component will automatically try to reach an optimal layout based on the current set of visible nodes. This separates it for most other graph visualization tools, which strive for an optimal layout, but do not allow for direct manipulation of the nodes.

Touchgraph also gives the user easy facilities for showing different “localities”, which are sub-parts of a graph. Users can interactively center the graph on one selected node, and the tool will search out a specified number of links from the selected one. This is extremely useful when examining a graph with thousands of points, because while constructing a readily comprehensible layout for thousands of linked nodes is very difficult, finding a good layout for twenty that are the focus of a view onto the graph is rapidly achievable.

The multivariate MST implementation presented here, called Linkgraph extends the Touchgraph tool described above, providing the mechanism for determining which nodes are linked in what way, while relying on the Touchgraph to perform effective graph layout and other graph appearance functions. Linkgraph combines the direct manipulation of graph nodes and attractive graph layout from the Touchgraph and integrates a VisualClassifier, the MST calculations, and coordination with other geospatial data visualization software components. The visual classifier, introduced in Figure 3-8, and shown with the layout used in the Linkgraph in Figure 3-34, enables classification and color choices based on those classes.

Figure 3-34: Visual Classifier component inside the Linkgraph
The relationships between the parts of the Linkgraph can be seen in Figure 3-35. The Linkgraph proper is a subclass, or descendent, of the Touchgraph class. Thus, the Linkgraph is able to inherit all the behaviors of the Touchgraph. In addition to this, the Linkgraph has references to the Visual Classifier, which provides classification and color scheme facilities, and to the NDimensionalMST, which performs the distance calculations. The NDimensionalMST in turn depends on a descriptive statistics class (for calculating correlations, ranges, etc.), and to an independent MST engine. Note that this independent engine could be replaced with another graph calculation method, without changing the user interface or other parts of the operations of Touchgraph.
Using MSTs for geospatial demographic data visualization

Figure 3-36 shows a simple example of how a large multivariate attribute space can be related to a single variable. The nodes again represent states; while the minimum spanning tree was found based on a selection of numeric representative long form variables, including race, income, and housing variables derived from 1990 Census data, normalizing by taking proportions. Color value is used to depict longitude, with a range from white (west) through dark blue (east). A glance at the graph reveals that states colored white are most often connected to other states colored white, and similarly with cyan colored states. This indicates, not surprisingly, that states that are similar in longitude are also similar in demographic composition (except for California and Texas). An additional video demonstration of the Linkgraph can be seen at [LinkGraph](https://www.touchgraph.com) [TouchGraph](https://www.touchgraph.com)
(http://www.geovista.psu.edu/members/hardisty/videos/linkgraph_1.avi). This tool provides a means of presenting data spaces to the user with interactive node-and-link graphs.

Figure 3-36: Minimum spanning tree of U.S. states based on thirty census variables

**MST Graphs for Variables: the Subspace Linkgraph**

A different, but complementary use for minimum spanning trees in the context of geospatial demographic data visualization is to create the minimum spanning tree on the variables in the data set, instead of the observations. Here, each variable is a node, and the distance from node to node is defined by the absolute value of the correlation coefficient of the two variables, or by any other inter-variable distance, such as conditional entropy (Patil, Myers et al. 2000). Then,
the same procedure for calculating which nodes are connected to which is followed as above, i.e. connecting first the closest nodes, then the closest unconnected nodes, then the closest node groups. Figure 3-37 shows the result.

This procedure is a means of automatically ordering the variables, trying to place similar variables near each other. It should be a particularly useful procedure when the analyst wants to explore a data set with many variables but does not know which variables are related to which. In addition, this procedure works to order variables in a way that can make their relationships more easily visible in other visual multivariate tools, by placing related variables near each other.

When a particular node is selected by clicking on it, it is highlighted in yellow, as POP1997 is in Figure 3-37. If the user clicks on the “Send Selection” button at the top, the set of variables that are in the current locality set, based on the maximum number of links shown from the selected variable, are broadcast to any interested listeners. The order is significant. The first variable is the selected one. This is considered the root of the tree. Then, recursively, the nearest node (or variable) is chosen, exploring each branch until each path has been taken. This procedure can be thought of as physically picking up one particular node, and seeing how all the other nodes dangle down in groups. In Figure 3-37, the proper order would be [POP1997, AVG_SALE87, AVG_SIZE87, BLACK, SEPERATED, NO_FARMS87]. AVG_SALE87 is second because it is the closest to POP1997, then AVG_SIZE87 because it is the next closest. BLACK follows because it is next on the current branch. Then, we have reached the end of a branch, so we backtrack to AVG_SALE87, then to the nearest un-traveled branch (SEPERATED) and so on. The advantage of doing this is that it maximizes the degree to which similar variables are placed near each other, facilitating visual analysis. Other procedures, such as ordering the nodes by their distance from the selected one, could also be used. The set of variables selected in this manner is referred to as a “subspace” (when considering the data as a large cube in n-dimensional space, as described above, any subset of variables defines a subspace). This
Subspace Linkgraph is an instantiation of the goal of automatically ordering sets of variables in a manner that places related variables near each other.

If all variables are selected, the subspace is identical to the full space. However, the Linkgraph has controls that allow the user to make visible only those nodes that are within one link of the selected node, or two, and so on. Using this facility, only a selected subset of variables defines the subspace.

When the user examines Figure 3-37, she can gain insights into which variables are close to each other in the overall data space. For example, the variable “HISPANIC” is closely related to the variables “OTHER” (for other races) and “AGE_UNDER5”, telling us that places that had
high proportions of Hispanics also had high proportions of people not identifying as any of the racial categories (or the inverse). This highlights the way that the U.S. Census has treated Hispanic ethnicity, making it parallel to the racial categories. The proximity of the variables “HISPANIC” and “AGE_UNDER5” informs the analyst that there is some strong relationship between age structure and Hispanic ethnicity. To further investigate these observations, the node labeled “HISPANIC” should be clicked on, then other listening components could show the data points, and potentially reveal the origin of the relationships.

Using Linkgraphs with Other Components

The power of using minimum spanning trees in conjunction with interactive graph layouts such as Linkgraph is further increased by pairing a Linkgraph with other components. For example, Figure 3-38 shows how a map might be used together with a Linkgraph. Clicking on a node in the Linkgraph causes the map to highlight the visible observations. Conversely, if one clicks on a location in the map, this will become the root of the tree that the Linkgraph shows. This allows the user to find geographic patterns starting from statistical ones. For example, in Figure 3-38, the Linkgraph is showing a group of U.S. States, centered on Washington State, and using thirty census variables about age, housing, race, and others. Washington State was selected in the map, and this selection was transmitted to the Linkgraph. Sending the selection from the Linkgraph to the map allows the user to quickly see that the states similar to (clustered near) Washington in this multivariate space are also clustered together in geographic space. The migration pattern that many people followed, of moving from Minnesota to Washington, is also reflected here. The Linkgraph is able to summarize a great deal of information, such as the thirty census variables used in this example, into a compact representation. The geographic patterns in the statistical clusters can then be explored in a linked map or other view. A video showing these
two components working together can be seen at

(http://www.geovista.psu.edu/members/hardisty/videos/linkgraph_map.avi).

Figure 3-38: Linkgraph paired with geographic map. Variables are sent by double clicking on the selected node. The variables being analyzed are thirty Census variables relating to race, housing, and farming.

Figure 3-39 shows how the Subspace Linkgraph becomes a more useful device if paired with a statistical visualization device like a parallel coordinate plot (discussed in detail previously in this chapter). In this configuration, when the button “send selection” on the subspace Linkgraph is clicked, the set of variables that are currently visible are sent, in an order that maximizes the degree to which like variables are near each other, to the parallel coordinate plot. This automatic ordering means that useful relationships can potentially be more easily extracted from the views in the parallel coordinate plot.
Conclusion

This chapter has integrated and extended ideas from three academic disciplines to address specific research challenges raised in chapter one for the design of geographic visualization for enumerated data, and how their application results in novel and potentially useful techniques and tools. The three academic disciplines are geographic visualization (geovisualization), Information Visualization (InfoVis), and Exploratory Data Analysis (EDA). Novel techniques resulting from the extension of concepts from these disciplines are categorized into direct geovisualization techniques, and those techniques relying on a combination of computation and visualization (dimension-reducing techniques).
For direct geovisualization, novel color anchoring techniques that build on cartographic traditions in systematic use of color to signify data relationships and the design and implementation for a flexible geographic mapping component are introduced. From InfoVis, spacefill visualization techniques are applied and combined with other geovisualization ideas to create a spacefill visualization technique that can be combined with maps and other visualization devices. From EDA, an Open Source parallel coordinate plot (PCP) called Parvis is extended to create a multivariate statistical visualization tool that enables symbolization and classification of polylines, in combination with direct manipulation of data axes.

Combining computation and visualization, a data reduction technique using minimum spanning trees (MST) is applied, and combined with interactive graph layouts. The resulting technique can provide the user with local views, of a complex data set, based on relationships in n-dimensional space. This exploratory technique can be combined with the coloring methods developed here to show relationships between a single variable and a set of variables.

Each of the techniques described above has been turned into a functional tool, in the case of the map and spacefill visualization, entirely written by the author of this dissertation, in the case of the node-and-edge graph visualization, leveraging an open source component called Touchgraph, and in the case of the PCP, leveraging an open source component called Parvis. All of these tools are designed to enable the demographic data analyst to visually understand highly multivariate data sets.

The mapping and color techniques and tools introduced in the direct geovisualization section each have novel and useful features for spatial data visualization. The color tools include a method for using color “anchors”. These anchors enable the user to easily and intuitively create multipart lines through color space. A color ramp picker incorporating these is encapsulated into a tool for matching classifications to symbolization methods. This combined component is called the Visual Classifier. The Visual Classifier is incorporated into a choropleth mapping component
that can be applied in a variety of contexts, including as a univariate map, a bivariate map, and inside matrix components.

The spacefill visualization component also has the Visual Classifier included in it, giving the advantage of a uniform interface to the user. The spacefill visualization component takes two variables for each display: one is used to color each observation, and the other is used to order each observation. This visualization component, like the map, has been adapted to fit in a matrix representation, making it a flexible component in this regard.

The PCP, extended from the open-source Parvis PCP, has (in common with other PCPs) the ability to show many variables at once. Where it goes beyond many extant PCPs is in the way it joins outside manipulation of axes, for ordering and conditioning, with the color and classification facilities of the Visual Classifier.

The section on combining computation and visualization using InfoVis techniques and associated tools is designed to enable data visualization of highly multivariate data spaces. These include a node-and-edge representation of a tabular data set called the Linkgraph. The Linkgraph enables browsing of local parts of a larger data set using a minimum spanning tree based on any subset of variables from the overall data set under analysis. This technique enables relating highly multivariate spaces to geography using the mapping component. The incorporation of a Visual Classifier into the Linkgraph enables comparison of any selected variable with the total data space represented in the Linkgraph, whether or not the variable is one of those included in the MST calculation.

All of these techniques joined together can help the demographic data analyst visually identify important patterns and outliers in their data. The techniques work together in that they all support a common set of events for communicating about important classification and symbolization choices the analyst has made. These events and the unified toolset, called the GeoViz Toolkit, are the subject of the next chapter, along with a detailed comparative analysis of
how these tools perform when applied to a data analysis problem related to geographic variation in cancer diagnosis and mortality.
References Cited


Chapter 4

Coordinated Geovisualization

This chapter explores the possibilities opened up by combining the coordination strategies described in Chapter 2, with the geo-info graphics described in Chapter 3, along with some additional geospatial demographic data visualization components. First, we examine different categories of coordination events (shared visual and numerical aspects of data representation) that are useful in geospatial demographic data visualization software. Next, we describe the elements of a coordinated geovisualization toolkit that uses the coordinated visual and numerical aspects of data representation, and how the components fit together in the geovisualization toolkit. Then, a case study based on analyzing health data from various sources, including the Appalachian Cancer Network, the Census Bureau, and the National Cancer Institute, shows the advantages and pitfalls of the toolkit.

This toolkit will be referred to as the “GeoViz Toolkit” in this chapter. A more complete name would be “The GeoViz module within GeoVISTA Studio, with associated classification, symbolization, and data transformation modules”. All components described here were written specifically to work well inside GeoVISTA Studio, and GeoVISTA Studio will be the primary distribution mechanism for this software in the future. Studio also provided the direct conceptual inspiration for the coordinator in the GeoViz toolkit. However, in this chapter, the toolkit is described as a stand-alone application, in part to separate the original contribution of the coordinator and applications described here from those in the core Studio engine and in part because it can work as a stand alone application for use without access to Studio.
This chapter addresses Goals 3.1, 3.2, and 3.3 of the dissertation. In relation to goal 3.1 (to find coordination categories) a specific objective here is to find what categories of visual and numerical properties are important to coordinate between pieces of geospatial demographic data visualization software, given a mechanism for coordination between components. Goal 3.2 (to populate the coordination categories with specific kinds of coordination) is addressed next. Goal 3.3 (provide a proof-of-concept demonstration) is met in two parts: first, by identifying some of the relative advantages and disadvantages of bivariate and multivariate tools, and second, by providing a proof-of-concept analysis session, showing the advantages of a coordinated analysis toolkit.

The coordination goals will be met by first describing two key previous projects in the field, which partially address the goals of the current research, and then by introducing in detail a set of visual and numerical aspects of data representation for coordinated geographic visualization, and a toolkit which uses them, developed as part of this dissertation research, the GeoViz Toolkit. The evaluative goals will be met by comparing the characteristics of bivariate and multivariate visualization components that are part of the GeoViz Toolkit. Both the description of the overall GeoViz Toolkit and the comparative evaluations of individual tools will be made in reference to a particular data set of interest. This data set reflects cancer mortality rates, screening rates, and stage of diagnosis, in a portion of Appalachia.

Meeting these goals highlights some of the contributions of this research, including the introduction of novel geovisualization methods that could be applied in other contexts and implementation of concrete tools that are available to interested analysts and tool developers. The contribution this portion of the research makes toward general strategies for design of geospatial demographic data visualization software are twofold. Firstly, the approach to delineating coordinated action and the software events that underlie them is general and applicable beyond the toolkit described, and beyond demographic data visualization. Secondly, the GeoViz Toolkit
provides a model integrated analysis environment that supports efforts to explore specific questions about effectiveness of this approach and provides an example against which other such environments can be compared.

The concrete contribution of this research is the functioning GeoViz Toolkit itself. This toolkit is freely available from (http://www.geovista.psu.edu/members/hardisty/). This can be used in its current form by end users, with different data sets, as long as the data set contains geographic data represented as polygons. The source code is available as a base that other researchers can build upon, available from the GeoVISTA Studio SourceForge web site.

The rest of this section discusses previously implemented integrated toolkits in geovisualization and related disciplines. These toolkits are considered because they have provided partial solutions to the problems posed here, and because the research presented here drew on the strategies advanced by these toolkits. This is followed in subsequent sections by a detailed discussion of the strategies for coordination used and components that embody these strategies in the GeoViz Toolkit. The individual components are then introduced, followed by a comparative analysis of the bivariate and multivariate techniques used. Finally, an integrated analysis session with an epidemiological data set demonstrates the kind of hypothesis generation that the coordinated toolkit enables.

**Integrated Toolkits: Background**

To understand the contribution of the GeoViz Toolkit, previous work in the area of geographic visualization which contributes towards the goals of identifying categories of coordination for geovisualization, and of creating a toolkit which uses them, needs to be described. Two important projects which make progress towards these exploratory geovisualization categories are the IRIS/Descartes/CommonGIS system, and the cdv project.
Both of these offer useful strategies that have been drawn upon in the current research. However, neither of them permits coordination with other components outside their system, making them “closed worlds”. Other integrated toolkits of note, which are less explicitly geo-exploratory, include Snap, Orca, and Polaris. Each is also discussed briefly.

The Descartes system, and its predecessor (IRIS) and successor (CommonGIS), while closed systems, are an excellent example of an integrated toolkit that enables hypothesis generation about geospatial data. The Descartes system is an Internet-based, exploratory geovisualization tool (Andrienko and Andrienko 1999d) that integrates EDA practices with geovisualization practices. This integration has great potential to “spatialize” the social sciences. There are many social science researchers in geography, sociology, political science, and so on, who deal with geo-referenced data on a daily basis. However, the geographic nature of geospatial data is not being fully taken advantage of (MacEachren and Kraak 2001). One potential solution to this problem is the development of tools that embody domain knowledge, like the knowledge of appropriate symbolization methods that cartographers and others have developed over time. Descartes implements just such a suite of tools. The graphic rules about the appropriate symbolization forms for different types of data are built in to the system; the user need not be aware of them.

Descartes was originally known as IRIS (Andrienko and Andrienko 1996; Andrienko and Andrienko 1997; Andrienko and Andrienko 1998c; Andrienko and Andrienko 1998b), and is now called CommonGIS (Andrienko and Andrienko 1999a). IRIS implemented the beginnings of some of the most interesting features of Descartes: integrating statistical and geographic presentations of the data, and incorporating current best practices into visualization methods. One goal that these features supported was that of distinguishing between exploratory and communicative goals in representations of geographic data.
Descartes incorporates methods to achieve the three core goals of geovisualization: allowing for private (i.e., individualized) map use, revealing unknowns, and supporting high interaction levels. Descartes also follows three important aspects of what comprises an EDA tool: linking, brushing, and focusing (see Chapter 3). However, Descartes goes beyond most EDA tools in its levels of geographic interactivity. For example, as Andrienko and Andrienko point out, in EDA tools implemented previously, the role of the map is only to carry information about spatial location. What Descartes implements is a map interface that incorporates multiple representations of the same data (Andrienko and Andrienko 1998a) and “makes traditional (cartographic) methods interactive and dynamic” (Andrienko and Andrienko 1999d), while making best guesses on appropriate presentation methods based on the characteristics of the data (Andrienko and Andrienko 1998c; Andrienko and Andrienko 1999a; Andrienko and Andrienko 2002). Descartes also integrates some data mining tools such as decision trees (Andrienko and Andrienko 1999c; Andrienko and Andrienko 1999b) and multi-criteria decision-support tools (Jankowski 1995; Jankowski, Nyerges et al. 1997; Jankowski, Andrienko et al. 2001).

Statistical tools available in Descartes include a Parallel Coordinates Plot (Andrienko and Andrienko 2001b; Andrienko and Andrienko 2001a). The PCP in Descartes is well integrated in terms of data classification and symbolization with the map component. Descartes can also represent data as a cumulative frequency curve (Andrienko and Andrienko 2001c) and has some facilities for visualizing temporal information, both as a statistical series (Andrienko, Andrienko et al. 2000a) and as moving objects (Andrienko, Andrienko et al. 2000b).

In sum, Descartes excels at providing integrated geovisualization functionality, with novel and useful user interfaces. Where it falls short of reaching the coordinated geovisualization goals highlighted here are in two ways: in the ad hoc nature of what kinds of visual and numerical aspects of data representation are coordinated between representations, and in the closed nature of their system. Descartes has added coordination of colors, of classification, and of selections, but
the coordinated aspects do not fit into any overall system of coordination. Descartes is a fairly closed world; it is difficult to add pieces from outside the system to Descartes, and, conversely, it is difficult to add pieces of Descartes to any other systems.

Another integrated geovisualization toolkit is the cdv project by Dykes (Dykes 1997; Dykes 1998). This software is designed to support ESDA (Exploratory Spatial Data Analysis). The system does so by incorporating some measures of local spatial autocorrelation, variograms, and other techniques. Cdv is a linked toolkit which presents a variety of different views, including scatterplots, maps, parallel coordinate plots, and cartograms, and links them together to highlight observations of interest. The parts of the program are loosely linked by sets of scripts, an approach which provides flexibility to the programmer. Like Descartes, cdv has an interesting set of coordinated functionality, but no overall plan to the coordination, and the parts of the program are limited in utility and extensibility because they are not independently deployable components.

There are a number of other projects incorporating integrated geographic visualization toolkits in other disciplines, although these are distinct from Descartes and cdv in that they are not as “map-centric”, and thus are less likely to be good models for coordinated geovisualization. These include efforts that link GIS and statistical packages (Cook, Symanzik et al. 1997; Swayne, Cook et al. 1998; Symanzik, Cook et al. 2000). The goal here has been to address a lack of support for exploratory visualization in GIS packages (Blaser, Sester et al. 2000). Part of the need that these efforts were directed at is now met by the Geostatistical Analyst extension to ArcGIS, does provide some exploratory tools in a GIS environment.

A significant advance in component-based coordination was made with the Snap system (North and Shneiderman 2000; North 2001). The Snap system allows any software which implements a simple COM (Component Object Model) interface to share selections between components. The operations behind Snap are similar to those of a database query, whereby
subsets of the data can be selected based on combinations of attributes. These connections can be of the one-to-one or one-to-many varieties, i.e. corresponding counties, or states to counties.

Orca (Sutherland, Rossini et al. 2000) is a visualization toolkit for high-dimensional data. It provides multiple linked views on data sets, including Snap-like brushing. It also includes facilities for time-series representations, parallel coordinate plots, and matrix-like representations. Unfortunately, the programming interface is forbidding enough that the project has been discontinued.

Polaris (Stolte and Hanrahan 2000) is a formalism for visualizing high dimensional datasets. Like Snap, it excels at helping the user define a query into a relational database. It also contains an interesting set of data-to-representation mappings, and geographic map representations (Stolte, Tang et al. 2002). The most novel feature of Polaris is a set of methods for visualizing and interacting with multi-scale data sets. Unfortunately, the system is not publicly available, so the coordination in the toolkit is difficult to assess. The written descriptions do not describe any coordination between different types of data views, just between scales.

The current research extends beyond advances of previous toolkits in two fundamental ways. The first is the broad range of interaction types that can be coordinated. The second is the fact that the coordination is based on automatic discovery of what the components support at runtime. There are aspects of both in the toolkits mentioned above, but none combine them.

Data selection is the most common kind of integration, included in all the toolkits mentioned above, but most do not go beyond it. The additional interaction types that the GeoViz Toolkit coordinates include classification methods and color schemes.

The integration methods used in the GeoViz toolkit mean that it is much easier to integrate components with different internal structures, which in turn means that it is much easier to integrate the work of other researchers into a common framework. The Linkgraph and the PCP (components modified from implementations by other authors) are examples of this in the
GeoViz Toolkit. This automatic discovery and coordination is also one way that the GeoViz Toolkit is an advance on both the Descartes system and cdv; those two systems are closed ones where all visualization components are expected to rely on the same data structures and in fact the same code base. The Snap project is open, like the GeoViz Toolkit, but it only supports coordination of selections.

Coordinated Visual and Numerical Aspects of Data Representation

In Chapter 2, the technical mechanisms and the general desirability of coordination was described. Here, we take a more detailed look at what coordination is to be done, as opposed to how to do it. We do this by first examining previous work in the area, then by going into some more detail about the mechanisms used to implement coordination in the current research, events and event listeners. Additionally, we will examine the rationale and data structures used for different categories of coordinated visual and numerical aspects of data representation.

Categories of Coordinated Visual and Numerical Aspects of Data Representation

Identifying the categories of coordination among types of visual and numerical aspects of data representation is a critical task for geographic visualization software for enumerated data. It is related to, but separate from, the task of identifying GIS operations (Albrecht and Kemppainen 1996; Albrecht 1999), space-time operators (Qian, Wachowicz et al. 1997), and interactivity types in geovisualization (Crampton 2002). Albrecht provides a set of universal GIS operations, including the following categories of operations: Search, Location Analysis, Terrain Analysis, Distribution/Neighborhood, Spatial Analysis, and Measurements. Crampton offers the following
four categories of interaction: (1) with the Data; (2) with the Data Representation; (3) with the Temporal Dimension; and (4) Contextualizing Interaction.

Albrecht’s and Crampton’s categories, while helpful, are in some aspects orthogonal to the categories of coordination being developed here. For example, coordination itself is considered a Contextualizing Interaction in Crampton’s typology, as is having multiple views on the data. Similarly, most of the operations in Albrecht’s work are specific to the spatial aspect of the data, aside from the “search” operation, which maps to selection. One of the conceptual bases for Crampton’s typology is a single, integrated map visualization component to which other components may be linked. The GeoViz toolkit was developed with the concept of co-equal components, none of which has a central role, except perhaps the coordinator itself. I draw on Crampton’s first three categories when defining my own (see below).

More open-ended, and therefore more applicable to the task of defining what categories of coordinated geovisualization are appropriate, is a taxonomy of visualization goals, presented in (Keller and Keller 1992). These “goals” are categories of visualization strategies, identified as pairs of two categories, an “action” and “data”. Types of action include: identify, locate, distinguish, categorize, cluster, rank, compare, associate, and correlate. Types of data include: scalar, nominal, direction, shape, position, spatially extended region, and structure. Thus, highlighting selected observations in a scatterplot would fall under the goal “identify cluster”, highlighting in a map would be “identify spatially extended region”. What is needed for categories of coordination is closer to the “action” categories, since different visualization components will have different “data” types that they operate upon, for example, the scatterplot and map operate on different sets of data, but may share “action” types.

Keim (Keim 2002) provides an interesting classification of visual data mining techniques that also has relevance here. He identifies data types (one-dimensional, two-dimensional, multidimensional, text, hierarchies, algorithms), visualization techniques (2D/3D plots,
transformed displays, icon-based displays, dense pixel displays, and stacked displays), and interaction techniques (interactive projection, interactive filtering, interactive zooming, interactive distortion, interactive linking and brushing). As candidates for types of coordination, the data types and the interaction techniques are both possibilities, and representatives from both are present in the categorization presented here.

Previous efforts to categorize operations and interaction focused on actions directed to a single application (or component). Those previous efforts to focus on coordination across linked views have (for the most part) ignored the work on kinds of operation and interaction, focusing primarily on selection operations. The present work addresses, directly, the problem of developing a framework that considers kinds of operation/interaction that can/should be coordinated in a multi-view, component-oriented environment. Thus, it addresses a different (and probably more challenging) problem.

It is most important to identify and support the aspects of visualization that are most widely shared across types of visualization component. Selection (discussed above) is expressible in nearly every component, and is thus an important construct to clearly conceptualize and support. Conversely, the space-filling visualization component introduced in Chapter 3 has, as one of its visual aspects, different fill orders (scan-line, spiral, Morton order), and these are not used by many components. In some situations, (such as when using two independent space-filling visualization components) it is not clear that coordination of fill order would even be desirable.

If we consider fill order from a higher level of granularity, we may want to coordinate it. Fill order can be conceptualized as an instantiation of the visual variable arrangement. An important distinction to make is between coordinating visual variables (red in one display = red in another display; last in order in one display = last in order in another display) and coordination of a data-to-variable mapping (the visual variable representing the highest category on one display =
the visual variable representing the highest category on another display – thus light red = first and
dark red = last). Both types of coordination are supported in the GeoViz Toolkit.

The following are tentatively proposed as primary types of coordinated visual and
numerical aspects of data representation: data, display, and category. These are arranged in
order of likely dependence in the construction of a geovisualization view. Data coordination is
the coordination of the set of entities under analysis. Data comes first, because it is the “universe”
that all other operations are applied to. Examples of data coordination would be applying the
same overall data set to a number of components simultaneously, and extending the data set to
include a derived field for each entity. Display coordination is the coordination of representation
methods. Examples of display coordination would be using the same data-to-display size
mappings in multiple components, and applying the same background color in multiple
components. Displays follow Data because the user may often wish to vary the symbolization to
better explore the data. Category coordination is the coordination of divisions (or groupings) in
the data. Examples of coordinated categories include linked brushing between components
(where categories are “highlighted” and “not highlighted”), and focusing on the same data range
in different components. Categories come last because they encompass such transitory operations
as which observation is being currently examined by the user. Each of these types represents
fundamental operations in geographic visualization that apply to many kinds of visualization
components. Each of these may also be expanded into subtypes. Below, the types and some sub-
types are expanded upon, and then the current set of events supported in the GeoViz Toolkit is
mapped onto these types.
Data

The “data” type includes events that carry the information that there is a new or different set of data to analyze. This kind of event indicates that the data space being analyzed has been changed. For example, if a spatial data set representing the provinces of Nepal replaces a data set consisting of the states of the United States, this should be communicated to any coordinated components. Extensions to the original data observations, including calculated fields or data linkages, would also be communicated using data type events. Similarly, extensions to variables, such as metadata on the origin and accuracy of different variables, would be communicated with data events.

Display

If the user has assigned some visual representation to some observations, these should be widely communicated and used. This coordination can enable discovery of spatial patterns based on non-spatial attribute data, and exploration of particular places in attribute data. Symbolization events could include information about many kinds of data to display mappings. Subtypes of symbolization include static visual properties, and dynamic visual properties.

Static visual properties useful for representing data identified by MacEachren (MacEachren 1995) include: location, size, crispness, resolution, transparency, color value, color saturation, color hue, texture, orientation, arrangement, and shape. A similar set was parsed by Wilkinson (Wilkinson 1999), following Bertin (Bertin 1981), into Form (size, shape, rotation), Color (hue, brightness, saturation), Texture (granularity, pattern, orientation), and Optics (blur, transparency). These subtypes (form, color, texture, optics) are a promising avenue to explore for coordinated symbolization.
Dynamic properties include variables applying to whole scenes, and to individual observations. Three "dynamic variables" – scene duration, rate of change between scenes, and scene order – were initially identified by DiBiase et al. (DiBiase, MacEachren et al. 1992), to which three more were later added: display date, frequency, and synchronization (MacEachren 1995). All of these can also be applied to individual observations as well, as could jitter and “motion paths” (Ware 2000).

Other important sensory modalities of information representation include sound (Krygier 1994) and touch (Griffin 2000). Further, data-to-display mappings and coordination of display form (e.g. a change from a choropleth map to a graduated circle map that should be reflected in all maps in a matrix) should be included here.

**Category**

Category type operations include many types of sub-setting operations on data sets. These are subdivided into “extent” types of coordination, and “classifying” types of coordination. The extent type includes attribute, spatial, and temporal, as further sub-types. The attribute subtype includes user-driven selections, as well as focusing and indication operations. Classifying includes such potentially coordinated aspects as traditional cartographic classifications, for example, selecting among quantile, equal interval, and minimum variation (optimal) classifications, and more complicated multivariate classifying, such as K-means clustering.

*Attribute Extent — Subsetting*

Subset coordination is the most common type of coordination among other coordination approaches such as (North and Shneiderman 2000) and (Hurley 2000), and has been used for
decades in statistical graphics (Fisherkeller, Friedman et al. 1974; Becker and Cleveland 1987; Cleveland 1987). Subsetting remains one of the most important types of coordinated event in a linked toolset. A user created subset is, almost by definition, a way that the user has of declaring, “these observations are of special interest”. Subsetting can be achieved through several types of action, such as brushing, indication, focusing, and conditioning. All of these action types produce subsets from a total collection of observations; in other words, they could be expressed using SQL “Select” statements.

Subspace changes, which are another type of subsetting, indicate that there is a different set of variables to analyze, consisting of some subset of the whole, for those analysis components capable of visually or numerically analyzing multiple variables. For example, if our whole data set had fifty variables, but we wished to examine six in more detail, this would be communicated as a subspace change. Subspace changes can also communicate the order that the variables should be presented in. This order is important because proximity of variables in tools like the PCP or matrix helps the analyst extract interesting relationships. One of the tools in the GeoViz toolkit, the Subspace Linkgraph, provides just such functionality. The utility of this is explored below in the analysis session section (Figure 4-36).

Brushing is subsetting driven by user interaction, as in “linking and brushing”. An indication is a transient subsetting (Raskin 2000), normally of just one observation. Indication events are normally triggered in a graphical user interface by a user pausing her pointer over a particular observation. This observation can then be separately highlighted in multiple representation components. Excentric labeling (Fekete and Plaisant 1999) provides multi-observation indication. Excentric labeling is described and illustrated below, in reference to Figure 4-4. Focusing signifies that a subset of the original data is being emphasized (either by entirely removing objects that are not within the “focus” of view or by otherwise changing appearance of the focus and/or non-focus objects). This differs from brushing in that brushing
highlights while focusing hides. *Conditioning* is the complement of focusing, but applied to a variable other than the one displayed. Conditioning, in the visual sense, consists of filtering the data set by some variable, and excluding from visibility those observations that lie outside the range for the conditioning variable. Conditioning can be used to explore relationships between correlated variables, seeing the remaining distribution in one variable when observations with a given data range in the other variable are filtered out (Carr, Wallin et al. 2000).

*Spatial and Temporal Extent*

Spatial, as a category, includes operations on the geographically referenced portion of a data set. These include panning and zooming operations, changing the field of view for the user. Both raster and vector types of spatial data operations need to be represented. Temporal extent would change the range of times that form the current view of the phenomenon under analysis. The user should be able to choose whether or not a temporal animation in one view coordinates with temporal animation in other views and if it does whether they are synchronized in time or offset in time.

*Classifying*

Classifying consists of categorizing the data based on some (often numerical) aspect of the data. It is a fundamental technique in geographic visualization (Mak and Coulson 1991; Egbert and Slocum 1992), in machine learning techniques (Whigham, McKay et al. 1992), and many other related fields. The *Journal of Classification* lists the following fields from which it solicits contributions:
statistics, psychology, biology, information retrieval, anthropology, archeology, astronomy, business, chemistry, computer science, economics, engineering, geography, geology, linguistics, marketing, mathematics, medicine, political science, psychiatry, sociology, and soil science (Heiser 2003)

Classifying is thus a critical aspect of geographic visualization to support across different components. One can, for example, classify the data using a self-organizing map technique, and see the results in a parallel coordinate plot and a choropleth map. The users’ choices for classifying are critically important information that should be leveraged by every component that can do so. In other words, if the user has arranged for the data to be grouped together on some basis, these groups should be widely communicated and used. The groups are the best guess that any component could have about what divisions in the data to highlight.

**Summary**

The framework outlined above characterizes the kinds of coordination that can be implemented in a multi-component, multi-view geovisualization environment (or InfoVis environment more generally), dividing kinds of coordination into data, display, and category types. The coordination mechanism described in Chapter 2 ensures that both the set of coordinated aspects of visual and numerical representation supported by any client and the total set of coordinated aspects can be added to incrementally. In other words, for each component, the number of coordinated aspects can be expanded over time, and their organization may be modified. At first, a component may only support common subsetting, but may add support for a common coloring scheme later, without breaking any kind of compatibility. Additionally, new coordinated aspects can be added as needed. So, if “fill order” was going to be coordinated, it could be added without interfering with the existing set of coordinated aspects of geovisualization component behavior.
Events: A mechanism for Coordinating Inter-Component Visual and Numerical Aspects of Data Representation

All of the coordination that takes place here uses the mechanism of Java events. Events, like their real-world counterparts, are “things that happen”. Events in Java were originally used exclusively for communication between graphical user interface (GUI) components. For example, a slider with a drag handle and a text box could communicate using events. Every time the user dragged the slider, the slider would fire an event, and the text box could receive it and update its internal state and appearance accordingly. However, recent versions of Java include a more generic event class, which supports communication between any components.

An important counterpart to the event is the event listener. An event listener is a class that implements an interface guaranteeing that the listening class will receive the event. With this bit of indirection, the concept of events and event listeners becomes much more flexible. Any class that wants to receive events from another class can register with the sending class. The sending class then need only keep a list of classes that have registered, and send events to these classes whenever the event trigger happens. Taking our previous example, a text box would have to implement the “change listener” interface to receive “change events” from the slider. Conversely, the slider could register with the text box to receive notification when the contents of the text box have changed.

All of the events used in the coordination system presented here are subclasses of the EventObject class. This class only has one method of interest, called getSource. This provides a reference to whatever class sends the event. Normally, events will provide access to additional methods that, in turn, allow access to some important data structures, either directly or as a callback to the sending class. For example, if a selection is changed in one class, then events will be broadcast that allow the receiving classes to find out what the current selection is.
What follows in the remainder of this section is a description of types of events that are coordinated in the GeoViz Toolkit. These are grouped into three main groups, as described above, data type events, display type events, and classifying type events. These follow the path from the most stable to the least stable, in terms of how often these events are fired in the GeoViz Toolkit. The events that are fired the least often represent the most stability from the application’s standpoint.

**Data type events**

In the GeoViz Toolkit, the only data type event that is currently implemented is the dataset event but more of these events will be supported in the future, such as data extension events. Dataset events are supported by almost every component, because the spatial and tabular data that are being visualized are carried in a dataset event. A dataset event carries a reference to a class, called DataSetForApps, which carries the data as a set of arrays.

**Display type events**

Symbolization can take many forms. The visual aspect component of visualization that is best supported in the current set of coordinated events is assignment of color. Two types of coordinated color events have been developed, one that was intended for components to straightforwardly assign a color to each observation, and another that encapsulated the information necessary to start from raw data and derive a classification and assign colors to the resulting classes.

The first approach, which is more passive on the part of those components receiving the colors, is appropriate for components that do not assign color themselves. For example, a parallel
coordinate plot could receive this type of color event and render each observation (thus each string) in the specified color. This coordination mechanism does not require any knowledge of variables displayed in the PCP; observations can be assigned a color based on any criteria (e.g., membership in a category for a variable not displayed). The data structure attached to the event is an array of colors, which has a length equal to the number of observations in the data set. The receiving component then merely needs to apply the colors to the observations when rendering, as illustrated in Figure 4-1.

![UML sequence diagram of the array of colors approach to passing color information](image)

**Figure 4-1:** UML sequence diagram of the array of colors approach to passing color information

The second, more encapsulated approach is appropriate for those components that base their coloring on their position in another component, as in the matrix and small-multiple components associated with the GeoViz Toolkit. The data structure passed is actually a class that can perform classification and coloring on the data, called in our system a ColorClassifier. The receiving component then passes data to this ColorClassifier and receives back a set of colors to
apply to the observations. Figure 4-2 illustrates some of the program flow involved in implementing this method.

The matrix component needs this kind of color information encapsulated in a display event because the matrix passes the symbolization information to its various elements without knowing what the coloring of each matrix element should be. One of the key capabilities of the matrix is that it does not impose symbolization, but enables each element to symbolize itself based on position in the matrix. For this kind of information to be passed from one component to another, some form of abstraction like the one shown in Figure 4-2 is necessary.

Figure 4-2: UML sequence diagram of reference to symbolizer approach to passing color information

Colors are tied to classifications in this scheme, but the coloring and the classification are separate objects. Thus, if future extenders of these components wish to combine variables like
shape or texture with classifications, the extension mechanisms should enable the reuse of the classification objects.

**Category events**

Category events include ones to define extents as well as events defining coordinated changes in classifying. The extent types instantiated include attribute and spatial events. The classifying events implemented include conditioning and classification.

*Attribute Extent – Subsetting*

Subset type events mean that some subset of the data has been singled out for special attention. In the GeoViz Toolkit, the names of the events that have been implemented and that fall into this category are selection (often by brushing), indication, subspace, and conditioning. Focus events are supported, via the mechanism of conditioning. The selection event itself is represented by an array of integers, which identifies the selected observations. This array is created by the broadcasting component, and read by all receiving components. The event carries a reference to the array. An example of coordinated selection is shown in Figure 4-3.
Indication events, in the current implementation, support a single observation. As noted, conceptually, indication can support multiple observations, as in excentric labeling. However, support for representing excentric labeling has been integrated into some of the components of the GeoViz Toolkit, but not into the indication event. Excentric labeling, for now, shows up only in the tool the user is interacting with, but the general model for coordination makes it possible for this event to be coordinated in the future without changing each component independently. Figure 4-4 shows an instance of indication in action between a map and parallel coordinate plot. The indication in the map is shown by a green hatching on the state of North Dakota. In the PCP, the string associated with North Dakota is highlighted. An example of excentric labeling is also shown (within the map only).
The subspace events consist of an array of integers, which give the indexes of the currently active columns of data to analyze. The order of the integers is significant; it informs the visualization components of the order in which the data variables should be displayed.

**Spatial type events**

The only spatial type event currently supported is the spatial extent event. This event type is broadcast by instances of geographic maps. It is useful to keep the same geographic area in view among multiple instances. The coordinator allows this to be turned on or off, so that it does not force all maps to show the same extent. As a result, it is possible to have both overview and detail (zoomed) maps in the same application. Statistical visualization components, like Parallel Coordinate Plots, can also potentially support these changes in extent. This would require either

---

**Figure 4-4:** Coordinated indication, with the indicated state set at North Dakota. Excentric labeling is also shown in the map, but not in the other components, because it is not (yet) supported by indication coordination mechanisms. Note that the indication is also reflected in the visual classifiers and in the bivariate legend.
making the PCP spatially aware, or would require adding information to the event, so that clients could find out which observations are within the spatial area. One difference between this type of event and a selection event is that it requires different symbolization in many clients; for example, a map might draw boundary lines with a wider stroke at a given extent. Other components may want to express different spatial extents with alternative symbolization as well, for example, when a map zoom causes a county-level depiction to change into a state-level depiction, the PCP might do the same.

Attribute extent – focus and conditioning

Conditioning is a filter on the dataset. If the filter is applied to the data variable currently being displayed, this is considered a focus event. If the filter is applied to another variable, this conforms to the usual definition of conditioning. Figure 4-5 shows a map, a PCP, and a device called the conditioning manager that provides conditioning facilities. Similar to the manner in which selection is implemented, conditioning is represented by a shared array of integers. The events broadcast the array to the listening components; the listening components receive the reference, and then read the array.
Classifying type events

Currently, there are two types of events that fall into the classifying type. One is the ColorClassifier event the other the classification event. ColorClassifier events are described in the previous section, because they fall into both the symbolization and classification categories.

A classifying event contains a reference to a classifier. A classifier, defined by a simple interface, is a class that can return an array of integers, when given an array of numeric data and a number of classes. This array of integers shows what class the given numbers fall into. For example, if a classifier is given the numbers \{23.4, 0, 2.1, 34.2\}, and asked to classify it into two classes, it might return the array \{1, 0, 0, 1\}, signifying that the middle two observations fall into the first class, and the first and last observations fall into a second class. This simple contract can
be fulfilled by many types of classification methods. Other Studio developers have implemented several such classifiers, three of which are used in the GeoViz Toolkit.

**Future additions**

In the future, the following are important event types that should be added. For all of these event types, the difficulty is less in finding an appropriate representation form for the event, and more in providing the necessary support in all associated components. Firstly, more types of subsetting should be allowed, i.e. intersection, xor selection, etc. Fuzzy sets would be a logical and potentially powerful extension. Secondly, events that describe an extension to the current data set should be included, i.e. a new attribute being added for each existing observation. Thirdly, symbolization should be expanded to represent size of symbol, shape, and other visual (and dynamic and sonic and haptic) variables as well as colors. Map algebra type events will be supported in the future. Additionally, temporal extent and temporal synchronization events need to be supported.

**The GeoViz Toolkit**

The GeoViz Toolkit consists of twelve freely available, open source, coordinated software components that are designed to support exploratory analysis of geographically referenced demographic data, such as population change data or health data. Some of these are extensions to components authored by other researchers. A more detailed description of authorship is provided at the end of this section.
1. Coordination Manager  
2. Spatial Data Import and Transformation Component  
3. Visual Classifier  
4. Univariate Map  
5. Bivariate Map  
6. Table Browser  
7. Parallel Coordinate Plot  
8. Link Graph  
9. Subspace Link Graph  
10. Map and Scatterplot Matrix  
11. Spacefill and Scatterplot Matrix  
12. Map and Spacefill Matrix  
13. Small Multiples Plot of Map, Spacefill, and Scatterplot  
14. Selection Animator  
15. Indication Animator  
16. Conditioning Animator  

Some of the components of this software have been described in earlier chapters, in particular, the Link Graph, Subspace Link Graph, and Spacefill visualization components. The remainder of this section will introduce new components and describe all of these components. Many components were introduced in detail in Chapter 3. Here, they will be briefly described, specifically focusing on the characteristics of these components for the analysis of the particular data set we will cover in detail, ACN (Appalachian Cancer Network) data. The section following the description of all components will compare and contrast what patterns can be found in the ACN data, systematically comparing what can be found in which components. This systematic comparison will use two key variables to compare bivariate components in their pattern visualization capacities, and six key variables to compare multivariate components in the same manner.

The components can be divided into four categories: foundational pieces that are required to do any analysis at all, univariate and bivariate visualizations, multivariate visualizations, and animators that send messages to the others. The foundational pieces include the Coordination Manager and the Data Import components. Without these, the data have no way of passing to all the components. The univariate and bivariate tools are the Visual Classifier, the Univariate and
Bivariate Maps, and the Table Browser. Multivariate tools include the Parallel Coordinate Plot, Link Graph, Subspace Link Graph, Map and Scatterplot Matrix, Spacefill and Scatterplot Matrix, Map and Spacefill Matrix, and the Map, Spacefill, and Scatterplot Small Multiples Plot. Finally, the Selection Animator, Indication Animator, and the Conditioning Animator do not display data themselves, but enable coordinated visualization by sending messages, corresponding to their names (Selection, Indication, and Conditioning) to other components. Figure 4-6 shows some of the GeoViz toolkit components showing some of the ACN data.

---

![Figure 4-6: Screenshot of five GeoViz toolkit components. Clockwise from the top left: PCP, map-scatterplot matrix, bivariate map, tablebrowser (mostly hidden), and Linkgraph.](image)

**Coordination Manager**

As described in Chapter 2, the Coordination Manager is a component that allows the user to flexibly coordinate software components in many configurations, and provides a user interface
for this coordination. The software automatically discovers what types of events, such as selection and coloring events, each component is trying to broadcast, and trying to receive. It then registers the components with each other. The optional graphical user interface allows the user to customize how the components are connected without doing any programming (Figure 4-7).

The functionality is provided via automatically discovered event listener interfaces, and event broadcaster methods. The pattern follows the standard set out in the Java language itself, in the EventSetDescriptor class, which is part of the Java core. In the official documentation on EventSetDescriptor, Sun describes a naming pattern:

the most simple standard design pattern where a named event "fred" is (1) delivered as a call on the single method of interface FredListener, (2) has a single argument of type FredEvent, and (3) where the FredListener may be registered with a call on an addFredListener method of the source component and removed with a call on a removeFredListener method. (Sun Microsystems 2003)

In other words, using selection as an example, there are three actors needed to follow this naming system, the listening component, the event sent, and the sending component. The naming convention that should be followed for the listening component is an interface that must be implemented, with the particular name, “SelectionListener”. The naming convention for the event itself is that it must be named “SelectionEvent”. The third convention is the form that the names of the methods that the sending component must have, “addSelectionListener” and “removeSelectionListener”. These methods are necessary to register and deregister the listening component from the list of interested components. Deviations from this naming strategy are possible, but only by using explicitly provided metadata in a companion class.

The Coordination Manager scans each component added to it for all interfaces implemented by the added component, and all methods implemented by the added component. It looks for matches that follow the naming pattern above. It then registers the components that are listening for particular events with those that are sending those events. A more detailed technical
description is provided in Chapter 2. The Coordination Manager also maintains lists of the sending and receiving beans so that the process may be reversed at a later time if needed.

If a GUI is desired, then a version of the Coordination Manager with an automatically generated user interface, shown in Figure 4-7, can be used. This component, named the CoordinationManagerGUI does none of the actual coordination, but instead is a shell around an instance of the non-GUI Coordination Manager. The GUI version passes along the components added to the Coordination Manager, and then queries the contained Coordination Manager for lists of the sending and receiving classes. The appropriate user interface parts for the GUI are then produced to support user control of coordination.

The foundational user interface element leveraged in the creation of the GUI is the check box. Each check box represents a one-directional connection between two components. If a box is checked or unchecked, that triggers a registration or deregistration between the two components involved. In the GUI, the organization is by listening component, and then sending component. In Figure 4-7, the first bean shown is the GeoMap. As labeled, this is the listening bean. Then, for each event type that the GeoMap is listening for, a set of check boxes is shown. The first set is DataSet events. The second set is Indication events. The third set is Selection events. Each of these sections is enclosed in a blue box and headed by the words “Event type:”. Following this, each component that is trying to send that type of event to the listener is listed. As mentioned, the first set is DataSet events. This contains only one sending component, ShapeFileToShape. The next event type, Indication, shows seven components that are trying to send indication to GeoMap. As shown, currently all connections are on. If any of the check boxes are unchecked by the user, this will cause the GeoMap to be deregistered for that event type and listener, and therefore not to receive this type of event. This process is much simpler to understand in action. Therefore, the reader is directed to (http://www.geovista.psu.edu/members/hardisty/videos),
where an example of the coordination manager graphical user interface being manipulated is viewable.

The significance of this kind of coordination is that it can be used to support automatic interaction between components for any kind of structures and events. Since we are most interested in geospatial demographic data visualization structures and events, the sets of events for selections, classifications, symbolizations, and so on, are used. There is no equivalent coordination system (known to the author), with the exception of GeoVISTA Studio itself, and Studio does not support end user control of coordination.

Figure 4-7: Graphical User Interface for the Coordination Manger
Spatial Data Import and Transformation Component

This is a set of mostly non-graphical components that import spatial and attribute data, and transform them into forms that other components can use. The data flow in the current GeoViz Toolkit is from a persistent store such as a file on disk, to components such as maps and parallel coordinate plots. First, a filename is chosen by the user. The chooser passes the file name to a data import class that creates a data table, including geographic features, and passes it along. Then, a map projection component steps in and projects the geospatial data. Then, it is passed along to any listening components. Figure 4-8 shows the icon associated with the projection component, because this is the component that is hooked into the coordinator. The power of this approach is that any equivalent data source may be substitute, and as long as it is able to fire DataSet events, the other components will not be affected.

Visual Classifier

Figure 4-9 shows the graphical user interface for the Visual Classifier, first introduced in Chapter 3. This component can stand alone, or it can be embedded in other components that use its facilities. As described in Chapter 3, there is a visual classifier currently embedded in the univariate and bivariate maps, in the parallel coordinate plot, the Linkgraph, and the stand-alone scatterplot. The visual classifier has three appearance modes, first, as shown in Figure 3-9,
entirely horizontal, second, in two rows, as shown in Figure 3-8, and third, in a vertical arrangement (shown below). The visual classifier tries to choose the best arrangement for itself based on the current space constraints in the component it is part of.

---

**Univariate Map**

The univariate map was introduced in Chapter 3. Recall that the univariate map has three basic constituent parts. From top to bottom, they are: a visual classifier, a map toolbar, and a map canvas. The visual classifier is the same as the one described above, using the two-level arrangement with a classifier picker on top and the color picker underneath, for a more efficient use of space. The ACN data set, as shown in Figure 4-10, poses no special problems for the map
component. Excentric labeling, as shown in the figure, is an aid in identifying the names of particular counties.

Figure 4-10: Univariate Map showing age adjusted cervical cancer rates

### Bivariate Map

The bivariate map interface (covered in more detail in Chapter 3 and shown in Figure 4-11) is the same as the univariate one with two exceptions: there is an additional classifier picker, and there is the addition of a bivariate color scheme legend as well. The meanings of the
choropleth colors shown in the map canvas are different than in the univariate map, however, because the bivariate map uses a bivariate color scheme, allowing it to present two variables at once.
Table Browser

The table browser, shown in Figure 4-12, gives a read-only view on the data set. This is a critical piece of the GeoViz toolkit for two reasons: information and reassurance. The informational part is that data values can be read off the table, including selections that can be
sent and received to and from the table browser. The reassurance part is that users who are not accustomed to using visualization components often feel “at sea” unless they have a tabular version of the data to fall back on. Even if the table is not relied on during an analysis session, its very presence gives many users the necessary confidence in the visualization tools to engage with them and attempt to interpret them.

Parallel Coordinate Plot

The parallel coordinate plot (Figure 4-13) was introduced in Chapter 3. Here, we mention how this component works in relation to the ACN data set. The ACN data set and this parallel coordinate plot are a good match in terms of the number of observations, but a less good one for the number of attributes for each observation. The number of observation is 156, which is a good number to show in the parallel coordinate plot without excessive over plotting. However, the ACN data set contains more than 50 variables by default, and trying to display all of these
simultaneously is impossible, or at least, would be quite visually cluttered. Thus, the PCP needs to be coordinated with a component such as the Subspace Linkgraph, or another variable selection component, to be useful in this context. Such coordination is discussed below.

**Link Graph**

The Linkgraph, described in detail in Chapter 3, is shown in Figure 4-14. This component’s capabilities are a good match with the ACN data set. The number of observations (156) can be comfortably handled inside the Linkgraph. The high dimensionality of this data set means that there is a need for the data-reduction facilities that this component provides.
Subspace Linkgraph

The Subspace Linkgraph is useful for exploring and automatically ordering a large number of variables, and for giving the analyst a route into the data, giving the analyst some overview information. The Subspace Linkgraph is also helpful when used in conjunction with tools like the matrix tools or parallel coordinate plot which rely to some extent on physical proximity between plots to make visual comparisons, because the Subspace Linkgraph provides an ordering that will help to reveal patterns in the data, as when the PCP places axes next to each other or when the multi-form matrices place matrix elements next to each other.

Figure 4-14: Linkgraph showing counties in the MST neighborhood of Centre county, using 24 variables relating to stage and age for cervical and breast cancer
Figure 4-15 shows the Subspace Linkgraph with ACN data. This component is helpful for undertaking an analysis of this data set because the dimensionality of the original data set is too high to directly represent in a parallel coordinate plot or other multivariate visual analysis component. By using the Subspace Linkgraph, and picking an appropriate starting variable, the user automatically can see other variables that have a strong relationship to the variable of interest.

However, the ACN data set also brings out the limitations of this component. Firstly, if there are a number of highly correlated variables, such as the numbers of lung cancers diagnosed at various stages, these will cluster together, and therefore the user will automatically see these grouped together, which might not be desirable, if these variables are previously known to co-vary. Secondly, Pearson’s correlation coefficient, which is used by the Subspace Linkgraph to order variables, will only identify linear (and aspatial) relationships. Thus, many types of close relationships between variables, ones that the human eye could easily discern based on a scatterplot or a bivariate map, may not be picked up by this tool. Thirdly, the relationships detected are all bivariate. If there is a strong relationship between three or more variables that does not exist between pairs of the individual variables, this will be missed by the Subspace Linkgraph.
Map and Scatterplot Matrix

Figure 4-16 shows a bivariate map and scatterplot matrix of ACN data. Both the position and the coloring of the observations is determined by the row and column position of each scatterplots’ position in the overall matrix, while in the map, position is geographic, and color is determined by the map’s positioning. Using this tool, we can see attribute and geographic relationships between multiple variables at a time. In this case, the variables are rates of cancer mortality for white females on each of four cancers. From top to bottom, they are breast cancer, cervical cancer, colon cancer, and lung cancer. We can use the scatterplots to read the colors in the maps, i.e. those observations that are red in a given scatterplot will also be red in the map that corresponds to the same two variables. We can see that cervical and lung cancers co-vary positively; there are more grays and whites than greens and purples in this map.
There is also a set of observations being shown as selected in all maps and scatterplots, with the same observations being selected in each. The observations selected were those that have high rates of breast cancer but low rates of cervical cancer. We can see in the maps that there is a high concentration of these cases in Pennsylvania. Thus, we can visualize the geographic element in this attribute data. In addition, we can see that these observations tend to be low in lung cancer and high in colon cancer by looking at that scatterplot.
Spacefill and Scatterplot Matrix

A spacefill and scatterplot matrix has some valuable properties when used with the ACN data, because (as noted in Chapter 3) the two types of graphs are complementary. An example of this type of display with 10 variables is shown in Figure 4-17. The primary advantage of having the scatterplot in this representation is that it is tied in with the spacefill component, and thus different aspects of distributions can be made evident. Selecting observations in the scatterplot and seeing where they show up in the spacefill component is an excellent way to understand how

Figure 4-16: Map and Scatterplot with ACN data
the spacefill component works. The advantage of the spacefill component is the good visibility of all observations. Over-plotting is avoided. For example, in Figure 4-17, the sixth and seventh variables shown (boxed in red) are both highly skewed towards low numbers. Thus, all the observations are clustered at the lower left-hand corner of the scatterplot.

In the corresponding spacefill visualization, a positive correlation between the variables is apparent (which is actually statistically significant), because the observations in the lower half of the spacefill visualization are lighter. Scan-line order is used here; order is by rows, starting from the lower left-hand corner, and proceeding leftward and upward. Thus, observations with lower values for the variable being sorted on will be lower down in the spacefill canvas.
Map and Spacefill Matrix

Figure 4-18 shows a matrix filled by using a map and a spacefill visualization component. This figure shows some of the strengths and weaknesses of this representation form for analyzing the ACN data. The strength of the bivariate maps in the matrix is that they can show geographic patterns quite clearly. In the case of the particular data being shown in Figure 4-18, there are dramatic breaks at state boundaries for most of these variables. This indicates that there is probably some bias in the way the data were collected that has been introduced at the state boundaries.
level. Take, for example, the map that is at the farthest lower right, which shows the bivariate relationship between “procto2ysm” (the proportion of men that have had proctologic exams in the last two years) and “smokecurM” (the proportion of men that currently smoke). The green scale is determined by the y axis, and the purple scale by the x axis, so, interpreting this map, white represents low rates of proctologic screening and low rates of smoking, dark gray represents high in both, greenish represents counties that have relatively higher rates of proctologic screening, and purplish represents those counties having relatively higher rates of men smoking. Note that Pennsylvania is clearly lighter, West Virginia greener, and Kentucky more purple. This shows that these variables have some strong geographic component, such a strong one, that one must suspect some systematic bias at the state level, rather than a true reflection of underlying rates. Such biases could include bias in data collection and interpretation, in provision and accessibility of health services, or in population characteristics and behavior.
Map, Spacefill, and Scatterplot Small Multiples Plot

The small multiples form of the matrix tool and sub-tools, shown in Figure 4-19, allows the user to see scatterplots, maps, and spacefill visualizations at the same time. This has some advantages, and one big disadvantage. The advantages are that comparative analyses can be done
using all three types of graph. For example, comparing current smoking rates for females with obesity, the scatterplot gives a good overall view of this positive relationship, the map shows the higher prevalence of both smoking and obesity in Kentucky, and the spacefill shows us that the places with higher obesity rates are clustered in places with higher smoking rates, but that places with lower obesity rates co-occur with places having both high and low smoking rates.

Figure 4-19: Small multiples matrix, showing scatterplot, map, and Spacefill components, with ACN data

**Selection and Indication Animators**

The Selection Animator and the Indication Animator demonstrate some of the power of coordination. They both were simple to create, because they leverage the existing event structures, and existing classification picker, combining them with a timer to do the work. Specifically, the components leverage the event sets that have been developed to give “grand tour” types of views of the data.

The selection animator works through combining the classifier picker component that is part of the Visual Classifier component, described above, with a timer and controls for the timer.
It broadcasts selection events based on the current classification, current variable in the classifier picker, and number of classes. If we have a data set with 800 observations and we set the animator to have 10 classes, first a selection with the 80 lowest observations for the current variable will be broadcast, then the next highest 80, and so on.

Similarly, the indication animator sends out indication events, which are given structure by organizing the order of the indications broadcast according to a classification. We take local stage of breast cancer as an example. If the data are broken up using classification into ten classes, based on the variable representing the proportion of breast cancer cases diagnosed at local stage in each county, then the indication animator will send out indication events for each county. The indication events would start with the county having the lowest proportion of cases diagnosed at local stage, and move up through each county in order, based on proportion diagnosed at local stage. There is an extra-long pause between classes, to give the user a sense of where these class breaks occur.

The user interfaces for the two components are quite similar, as Figure 4-20 shows. Along the top, there is a start/stop button, a “subspace” checkbox, and a slider to give the user control over pace. The start/stop button starts or stops the animation. Whether the components are in the start or stop states has no bearing on the other functions, in other words, the components remain interactive during animation. The subspace checkbox indicates whether the component should iterate, variable-by-variable, over the last sent subspace (set of variables), or just iterate over the currently selected variable. For example, with the subspace box checked, if another component sent a subspace consisting of four variables, both the interaction and selection animators would iterate over each of the four variables, moving from low to high in the first, in the second, and so on to the fourth, and then cycle back to the first and continue. Along the bottom portion of the interfaces, a classification picker is inserted. This is the identical classification picker that forms a part of the visual classifier, shown in Figure 4-9. There is a two-
way communication between these animators and their classifier pickers. If the user alters the
classification by moving the slider bar that controls the number of classes, or the classification
type, or the drop-down list that shows the current variable under analysis, the overall component
picks up that change and starts broadcasting on that basis. Alternatively, as the animators iterate
over different variables in either a subspace or the entire data set, the current variable is
communicated to the classifier picker and displayed there. A selection animator in action can be
seen at (http://www.geovista.psu.edu/members/hardisty/videos/pcp_sel_anim.avi).

---

**Conditioning Animator**

The conditioning animator, shown in Figure 4-21, is a closely related component to the
above two animators, except that it broadcasts conditioning events instead of selections.
Conditioning means excluding from consideration some observations, often based on a different
variable than the one being used in classification and symbolization. The conditioning animator
has a simpler interface that the indication animator and selection animator discussed immediately
above. The start-stop button has the same function. Next, there is a drop down list that shows the
currently selected variable. Next, there are two radio buttons showing the current choice of
animating over all variables in the current subset, or just animating over the current variable.
Finally, there is a slider that controls the pace of the animation.
The GeoViz Toolkit is built upon code from many authors, and it is important to make the contribution of the various contributors clear. My contributions came in several forms: the coordinator and associated mechanisms, some entire components (particularly the mapping components), collaboration on other components (particularly the matrixes), and adaptation for coordinated use of still others.

In the case of the PCP and Linkgraph components, I took what were fully functional components on their own, and adapted them. These were Open Source components (see Chapter 2), making their adoption, modification, and re-release possible. They both had the weakness that they were designed for stand-alone operation, not for co-existence with other components. Thus, they had to be changed to accept a common data format. Similarly, they did not have classification or symbolization facilities built in, so these had to be added.

Taking the list of components that have been introduced thus far, here is an attribution, with the authors’ names in order of the primacy of their contribution.

1. Coordination Manager – Frank Hardisty
2. Univariate Map – Frank Hardisty
3. Bivariate Map – Frank Hardisty
4. Selection Animator – Frank Hardisty
5. Indication Animator – Frank Hardisty
6. Visual Classifier – Frank Hardisty, Xiping Dai
The importance of Java as an underlying platform should also not be overlooked. The components listed here all leverage the libraries of code that come with the Java runtime. Such problems as how to do screen painting, graphical user interface creation, and data import are all greatly eased by the presence of Sun’s Java APIs (application programming interface). Another important aspect of Java from this perspective is the fact that the whole source code for the platform is published and released with every installation of the Java Development Kit (JDK). This is invaluable for both understanding the behavior of Java code, and for looking for solutions and design patterns when creating Java classes that depend on Sun’s classes. Java may not be open source software, but it comes close. Having access to read (if not to change) the internal structure and behavior of all components is a substantial benefit for constructing such components.

Relationships Found in an Epidemiological Data Set: A comparison of bivariate and multivariate tools

Before conducting an integrated analysis, it is helpful to understand what types of observations the various pieces of the toolkit are better at helping the analyst make. To understand the types of hypotheses that the tools in the GeoViz Toolkit may generate, a comparative analysis of the bivariate and multivariate tools is performed. Each tool is systematically examined for the possibility of recognizing the presence of exceptional particular observations, or outliers, and the possibility of identifying overall patterns, both spatial and non-
spatial. This systematic examination is performed using a dataset identified as important by the Appalachia Cancer Network (ACN).

The ACN is one of seventeen special populations networks funded by the National Cancer Institute. Appalachia is singled out for special attention, because it is a medically underserved area (Friedell, Rubio et al. 2001), and it has persistently higher rates of some cancers, including cervical and breast cancer (Hall, Rogers et al. 2000). The ACN data set used here is an amalgam of data from the cancer registries of the states that are participants in the network, combined with BRFSS data (explained below). In particular, the proof-of-concept application of GeoViz Toolkit data exploration methods presented below focuses on the Appalachian counties that are considered gold registries by the North American Association of Central Cancer Registries (NAACCR), which means they have high standards for the completeness and accuracy of their data, (better than 98% completeness). These counties are in three states, Pennsylvania, West Virginia, and Kentucky. One of the interesting aspects of NAACCR data is that they include statistics on the stage of a cancer within the human body at the time of diagnosis. There are four stages: local stage, regional stage, distant stage, and un-staged (stage data missing). Local stage means the cancer was still localized when it was detected; this gives the best prognosis, and is the most desirable. Regional means that the cancer has spread outside the organ that it started in, often to the nearest lymph nodes. Distant means that the cancer has spread to other parts of the body.

One of the particular interests of medical researchers is to compare patterns of cervical cancer with those of breast cancer. To this end, we will examine bivariate and multivariate tools comparatively. NAACCR data have been combined with age-adjusted rates for these two cancers, provided by the National Cancer Institute. Another two variables, which provide information about screening rates, are taken from a telephone survey called the Behavioral Risk Factor Surveillance System (BRFSS) conducted by the Centers for Disease Control (CDC). The two
BRFSS variables are the proportion of women age 50-64 who have had Pap smears within the last three years, and the proportion of women age 50-64 who have had mammograms within the last two years. It is important to recognize that the sampling scheme for this survey was designed to do state-level comparisons, so that although county level data are available for some variables the measures for specific counties cannot be considered reliable.

A starting assumption in the analysis is that these data will be closely related, in two groups. Rate of detection of cervical cancer at local stage should be positively correlated with the proportion of women who have had Pap smears within the last three years, and negatively correlated with mortality rates, because Pap smears are the means of diagnosing cervical cancer, and early detection should lower mortality, and in the case of this cancer can even prevent cancer since there is a pre-cancerous condition that, if treated, will not progress to cancer. Similarly, proportions of breast cancer diagnosed at local stage should be positively correlated with proportions of women having had mammograms within the last two years, and inversely correlated with breast cancer mortality. However, the actual statistical picture is not so clear cut, as shown in Figure 4-22, which is a table showing the correlation matrix for these six variables.

<table>
<thead>
<tr>
<th></th>
<th>Mammograms</th>
<th>Pap Smears</th>
<th>Local Breast</th>
<th>Local Cervical</th>
<th>Breast Mortality</th>
<th>Cervical Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mammograms</td>
<td>0.38 &lt;0.01</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
<td>&lt;0.01</td>
</tr>
<tr>
<td>Pap Smears</td>
<td>0.16 0.22</td>
<td>0.04 0.04</td>
<td>0.06 0.06</td>
<td>0.06 0.06</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>Local Breast</td>
<td>0.15 0.23</td>
<td>0.23 0.23</td>
<td>0.23 0.23</td>
<td>0.90 0.90</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Local Cervical</td>
<td>0.15 0.24</td>
<td>-0.01 -0.01</td>
<td>-0.01 -0.01</td>
<td>0.90 0.90</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Breast Mortality</td>
<td>0.19 0.24</td>
<td>0 0</td>
<td>-0.01 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4-22: Pearson’s correlation coefficients (lower left) and two-tailed t scores (upper right) for ACN data set
Of the hypothesized relationships, only the positive relationships between screening and diagnosis at local stage exist as statistically significant correlation coefficients (breast, \( t = 0.04 \), cervical, \( t < 0.01 \)). There is a surprising positive correlation between the rates of screening and the rates of mortality (breast, \( t = 0.06 \), cervical, \( t < 0.01 \)). Further, there is no relationship between the proportion diagnosed at local stage and mortality rates. Many factors could contribute to the failure of observed relationships to match expectations. Foremost is the fact that the mortality data are age-adjusted, while the other variables are not. Therefore, places that have higher age-adjusted screening rates might well have lower mortality rates, but we have no way of knowing that. Secondly, correlation is a limited tool, only linear correlation was applied, and other important relationships besides linear (aspatial) relationships might well exist in the data. The goal here is not to create a complete data analysis environment that supports exploration, hypothesis generation, and hypothesis confirmation, but to develop coordinated visualization methods and tools to support the first two stages. Related research that I am collaborating on is adding the statistical methods necessary to do a more comprehensive analysis.

**Bivariate Tool comparison: Bivariate map, scatterplot, spacefill visualization**

The tools are compared using two key variables from the ACN data set: proportion of breast cancer cases diagnosed at local stage, and proportion of cervical cancer cases diagnosed at local stage. Cervical cancer is of particular interest for two reasons: it occurs at relatively higher rates in Appalachia compared with the nation as a whole, and it is considered an almost completely preventable cause of mortality. The goals of this section are to critically assess the components, both to compare them against each other, and to make recommendations about the effectiveness of different tools for geospatial demographic data visualization. The criteria used to assess these tools are first, ability to identify statistical outliers, second, ability to identify
geographic outliers, third, interesting statistical patterns noted, and fourth, interesting geographic patterns noted.

*Bivariate: Bivariate map*

Figure 4-23 shows the bivariate map tool, mapping the proportion of cervical and breast cancer at local stage. Each variable is divided into three equal quantiles, with each quantile containing the same number of counties. The counties that have relatively high rates of these cancers being detected at local stage (the best result) are shown in dark gray. Conversely, those counties with relatively low rates of cancers being diagnosed at local stage (the worst result) are shown in light gray. Those counties with a higher rate of breast cancer diagnosed at local stage than cervical cancer at local stage, relative to other counties in the study area (which would imply that there is a potential problem with cervical cancers being diagnosed) are shown in green. Those counties with a higher rate of cervical cancer diagnosed at local stage than breast cancer, relative to other counties in the study area, are shown in purple.

Considering the four diagnostic criteria, firstly, the facilities for showing statistical outliers with this tool are not good. This is partly a function of the classification and symbolization choices available in the tool; but it also reflects the tool itself, which emphasizes geographic distributions over the statistical ones. Secondly, the analyst can see some geographic clusters using this tool. There are groups of counties with lower proportions of the target cancers being diagnosed at local stage, in particular, in northeast Kentucky and northeast Pennsylvania. Thirdly, the analyst can see some overall statistical patterns, via the bivariate color scheme. For example, since there are fairly large numbers of greenish and purplish colored counties, as opposed to gray ones, the analyst can deduce that these two variables are not highly correlated, and that whatever local correlations are observed, they do not create a pattern that is significant
overall. Fourthly, we can see some interesting overall geographic patterns with this tool. Pennsylvania is notably greener than the other two states, indicating that it may be able to improve its relative performance in detecting cervical cancer at local stage.
Figure 4-24 shows a bivariate scatterplot with the same data, showing the proportion of cervical and breast cancer at local stage. The color scheme is the same as in the map above, with low-low rates in light gray, and high-high rates in dark gray, with greenish and purplish.
signifying lower proportions of cervical and breast diagnoses at local stage, respectively. The color information largely encodes the same information as the location of each dot on the scatterplot, except that the location of the dots is on a linear scale, while the coloring is determined by the quantile position of each observation.

The qualities of the scatterplot as a visualization tool are the inverse of the bivariate map, as might be expected. The scatterplot shows some important anomalies; perhaps most important, in this pair of variables, it alerts the analyst to the fact that there are a number of values with 0 percent of cervical cancer cases diagnosed at local stage, and a number with 100 percent diagnosed at local stage. This tells us that the percentages are based on small numbers, and may not be a reliable base for statistical analysis. The scatterplot contains no geographic information, and so can give us no direct information about spatial outliers or spatial patterns. When coordinated with a map, coordinated classification and symbolization choices help to rectify this. Overall, these two variables do not appear to have any striking relationship in the scatterplot, aside from a higher density of values at the 60 – 75 percent range for breast cancer, and 40 – 70 for cervical cancer.

One difficulty with this popular representation form is that there can be multiple observations with the same value, which would form an important cluster of values to understand, but there would be no way of detecting this situation. One means of surmounting this problem, which is implemented in the GeoViz Toolkit, is to use “grand tour” selection, and note when there is a fall in the number of visible selected points. Another means, also implemented in the GeoViz toolkit, is to employ excentric labeling. Even if several observations occupy the same point in the graph, they will be revealed by multiple labels with lines connected to that point. This is demonstrated in the figure below, with Cumberland (KY), Clinton (PA), and Northumberland (PA) appearing as one dot on the scatterplot, but clearly identified using excentric labeling.
Bivariate: spacefill visualization

Figure 4-25 shows a spacefill visualization representation of the same two variables, the proportion of cervical and breast cancer at local stage in 156 Appalachian counties. Unlike the
two previous representation methods, this does not rely on bivariate coloring. In this visualization device, one variable is used to color the observations, while the other is used to order them. In this case, the observations are colored by percentage of breast cancers diagnosed at local stage (with dark purple still representing high breast cancer %), and ordered by the percentage of cervical cases diagnosed at local stage.

The spacefill visualization component suffers from the same lack of spatial display capabilities that the scatterplot does. It also fails to show a picture of the data that is as immediately interpretable. However, as discussed above, it can show patterns that other representation forms do not. In this case, we see that there is a large cluster of darker observations in the middle of the plot area, with the lightest valued observations towards the top and bottom. This shows that there is a bi-modal distribution in the relationship between breast cancer and cervical cancer: those places with a low proportion of breast cancer cases diagnosed at local stage tend to be either high or low, but not in the mid-range, for diagnosis of cervical cancer at local stage. Conversely, those places with high proportions of breast cancer diagnosed at local stage tend to cluster in the mid-range for diagnosis of cervical cancer at local stage. These relationships can be seen in the scatterplot as well, but not as clearly.
Bivariate: Linkgraph

Figure 4-26 shows part of the Linkgraph representation, using proportions of cervical and breast cancer diagnosed at local stage to calculate the MST. Like the other bivariate representation types, it cannot reveal spatial patterns. However, it can identify clusters that other forms do not as easily reveal.

Figure 4-25: Spacefill visualization of proportions of cervical and breast cancers diagnosed at local stage in NACCR gold registry states within the ACN. The highlighted regions show how places that have a low proportion of breast cancer cases diagnosed at local stage tend to have either high or low proportions of cervical cancer diagnosed at local stage.
In Figure 4-26, in the middle-right of the figure, the counties Northumberland, Clinton, and Cumberland can be seen in a tight cluster. This particular cluster is not well shown in any of the other bivariate forms, except by using excentric labeling, for which the user must manually search the whole analysis area. The map divides the statistical space so roughly that the similarity of these counties is overlooked. In the scatterplot, they are missed because they are, in fact, mostly on top of one another. In the PCP (shown below), because these three counties’ values are in the mid-range of the values for both variables, it is difficult to spot them as a local cluster. In addition, the Linkgraph textual representation allows for distinct recognition of each county. This is a large advantage over the spacefill representation, particularly for this size of dataset.

Figure 4-26: Linkgraph visualization of proportions of cervical and breast cancers diagnosed at local stage in NACCR gold registry states within the ACN
The Linkgraph property of being able to identify local clusters is even more pronounced using other variables. In Figure 4-27, the MST for mammogram and Pap smear rates is shown. There are a number of tight clusters here, which do not show up well under other representation forms. These tight clusters are clusters of more urban and more rural counties, perhaps accounting for their similar screening rates.

Multivariate tool comparison: Parallel coordinate plot, Linkgraph, matrixes

For this comparison, the six key variables introduced above, from NAACCR, NCI, and BRFSS will be examined. These six variables are proportions of cervical and breast cancer
detected at local stage, age adjusted cervical and breast cancer mortality, and proportions of women who have had Pap smears within the last three years and mammograms within the last two years. They are of interest especially because any interrelationships between them will be helpful in deciding where and how potential interventions to improve screening procedures might improve cancer outcomes. The criteria used to assess these tools are (1) ability to identify spatial and statistical outliers, (2) interesting statistical patterns noted, (3) interesting geographic patterns noted.

The PCP and matrix-based tools are capable of responding to Subspace events, which are events that define a group of variables, and the order in which they should appear. Figure Error! Reference source not found. shows the Subspace Linkgraph component, featuring the ACN data variables being used in this comparative analysis. As noted above, the Subspace Linkgraph orders the variables in a way that traverses them while minimizing the distance between all nodes, defining “distance” as one minus the absolute value of the correlation coefficient. This technique attempts to group variables that are more closely related to each other. In the PCP, the angle of the lines and whether or not they cross is relevant only between adjacent axes; in the matrices, relationships between adjacent pairs of representations are more easily compared.

In these comparative tests, the variable order that the Subspace Linkgraph found and transmitted is the same for all the components: mamog2ysm, pap3yrsm, cervical_mort_age_adj, per_allagecervlocal, per_allagebreastlocal, breast_mort_age_adj. This ordering is interesting in itself, because it groups together the BRFSS data and the ACN local stage, and because the cervical cancer variables, on screening, mortality, and stage of diagnosis, are grouped together, as are the breast cancer mortality and local stage variables.
Multivariate: Parvis PCP

In this example analysis, the polylines that represent counties in the extended Parvis PCP have been colored according to the age adjusted cervical cancer mortality variable (Figure 4-29). Counties with the lowest mortality rates are shown in black, counties with high mortality are shown in light purple, and the counties in between are shown with a color that is between the two (dark purple). The mouse pointer is over Calhoun County, so this polyline is indicated.

The first criterion on which to evaluate the PCP is by success at outlier identification. The PCP as demonstrated here is an effective tool for this task. The Parvis PCP has an advantage over most other PCPs in this regard due to its use of translucency. Translucent representations are partially transparent. If multiple translucent polylines are laid on top of one another, they are
darker than single polylines that do not have any overlay. Using the data set shown below, one such set of overlying polylines that shows an outlier is the set of lines that have an age-adjusted cervical mortality rate of zero, but have 100 per cent of cervical cancer cases diagnosed at local stage. These observations show up as a dark line between the bottom of the third axis to the top of the fourth. The analyst can also observe a set of outliers that have zero mortality and zero percent of cervical cancers diagnosed at local stage.

In terms of the overall statistical patterns that can be detected using this tool, one interesting pattern is an indistinct band of the lightest purple polylines that runs through the last five axes. This band can be further investigated by selecting on it, making those polylines darker in the PCP and showing whether they have any spatial patterns in geographic representations. A pattern that is significant by its absence is the strongest bivariate pattern in the data set, as measured by correlation coefficients, the positive correlation between mammograms and Pap smears. It is difficult to see this pattern in Figure 4-29. Changing the classification to match either of these two variables, or changing the scaling of the mammogram variable to match that of the Pap smear variable helps this pattern to emerge more clearly.
Multivariate: Linkgraph

The Linkgraph representation shows counties clustered as measured by their linkages in a minimum spanning tree based on a set of variables. Figure 4-30 shows the tree generated by using all six variables in the data set, and coloring by the Pap smear variable. For finding outliers, this representation form has significant advantages and disadvantages. The significant advantage is the ability to view all six variables at once. Counties that are particularly far from all other counties in this representation, such as Wayne County in the middle-left of Figure 4-30, are
outliers in the total, six dimensional space. Why this county is an outlier, and what it means in the total context of the data set, this representation cannot answer by itself, but it is the only tool in the GeoViz Toolkit that would detect it as an outlier. Following up with Wayne County using the full GeoViz Toolkit reveals the reason for Wayne County’s uniqueness. Firstly, Wayne has null data for the mammogram and Pap smear variables. This groups it with the other five counties that have nulls for these two variables. Among these seven counties, Wayne is middle-high for cervical mortality, which none of the other observations are. The tool that made these observations possible is the space filling one, because it has the best null visualization facilities.

One weakness of the Linkgraph as an outlier detector is its unreliability for detecting all outliers. If an outlier sits between two clusters, “chaining” will cause it to be connected on both sides, and therefore difficult to spot as an outlier. Additionally, the implementation of the MST in n-dimensional space handles nulls by treating them as the mean for that variable, which hides the null values from view overall, although it makes them similar to each other. Ideally, null values would be distinct, without destroying the graph.

Similarly, for overall patterns as for outliers, the Linkgraph representation can detect patterns that other tools may not, but will fail to show patterns that are obvious in other tools. Using the drop down list in the visual classifier makes it easy to cycle through the variables and look for patterns. In Figure 4-30, there is a striking pattern between the colorings and the linkages formed by the overall data set, reflecting a strong relationship between the rates of women who had Pap smears within the last three years, and the full set of variables. The connected counties along the top of the figure are all in the category of counties in which a low proportion of women have had Pap smears within the last three years. This variable, then, provides a “cut point” that can distinguish between counties in the overall data set.

On the other hand, there are bivariate relationships that are hidden by this representation. For example, there is a positive correlation between the proportion of cases that are diagnosed at
local stage for cervical cancer and those for breast cancer. This important relationship cannot be easily detected in this Linkgraph representation.

Figure 4-31 shows a map and scatterplot matrix representation of the data set, with the representation ordered in the order found by the Subspace Linkgraph (mamog2ysm, pap3yrsm, cervical_mort_age_adj, per_allagecervlocal, per_allagebreastlocal, breast_mort_age_adj). This

**Multivariate: Map and scatterplot matrix**

Figure 4-30: Linkgraph showing 6D space: with proportions of women having had mammograms and Pap smears, along with both mortality rates and proportion diagnosed at local stage for cervical and breast cancer. Color represents the proportion of women who have had Pap smears within the last three years. The strong relationship between the link patterns and the color patterns shows a relationship between the Pap smear variable and the overall data set.
representation is a good complement to the Linkgraph representation discussed above. It does not show relationships between all variables at once the way the Linkgraph does. On the other hand, it can show bivariate statistical and geographic patterns very well.

Outliers, both statistical and spatial, can be seen in this representation. The scatterplots can show groups of data points that are far away from the main mass of data, as in the counties that have zeros recorded for them in the mammogram variable, seen along the top of Figure 4-31. Spatial clusters can be seen, for example, there is a cluster of counties in West Virginia that have high rates of women having had Pap smears in the last three years, and having high age-adjusted cervical mortality rates. This cluster would be worth examining in more detail to see why this counter-intuitive relationship exists so strongly here.

The bivariate color schemes used in the maps bring the advantage of giving the analyst the ability to visually scan the maps and look for both statistical and spatial relationships. The statistical relationships are shown by the amount of color in each map. Those maps that show two variables that positively co-vary will have more light and dark grays. Those maps without strong relationships between the depicted variables will have more color. For example, the positive relationship between rates of mammogram and Pap smear screening in the map in the upper left has clearly discernable light and dark gray patches, which shows both local spatial clusters, as well as a global covariance. By contrast, the map closest to the middle of the representation, which shows the relationship between cervical cancer mortality and the proportion of cervical cancer cases that are diagnosed at local stage, has more greens and purples, signaling its weaker relationship.
Figure 4-31: Map and scatterplot matrix showing proportions of women having had mammograms and Pap smears, and mortality rates and proportion diagnosed at local stage for cervical and breast cancer. The upper dark cluster shows an area with high rates of both kinds of screening. The lower two lighter clusters show areas of relatively higher cervical cancer mortality (green) in relation to proportions of cancer diagnosed at local stage (purple).

**Multivariate: Scatterplot and spacefill matrix**

Figure 4-32 shows a matrix of scatterplot and spacefill plots. The same variables are shown, but the rate of Pap smear screening has been moved to the end, to show one of the capabilities of this representation form. The scatterplot and spacefill representations are much
closer cousins than the map and scatterplot or map and spacefill; this reduces the utility of this representation form somewhat.

Two important capacities that this spacefill representation form has that the scatterplot does not are showing multiple observations that have the same values, and showing null, or missing, values. The spacefill representations along the bottom are colored by the Pap smear variable, which has seven null values. The null values are colored dark gray. The location of these null values can be seen clearly in relation to the distribution of the other variables, for example, when compared with the breast cancer mortality variable, there are relatively few nulls in the counties that have the lowest breast cancer mortality rates. This may be important information to highlight when trying to understand the bivariate relationship between the two. The spacefill at the bottom left corner shows a solid strip of nulls. This strip appears because spacefill representations place nulls at the end for the ordering variable.

The scatterplot, by contrast, does not display nulls, instead simply omitting them. This problem could be overcome by adding a “null” value on the axis either at the top or at the bottom of the axis. Even if this were done, the seven observations that are null in both the Pap smears and mammograms would appear as a single dot in a scatterplot, whereas they appear as seven distinct squares in a spacefill.
Figure 4-32: Spacefill and scatterplot matrix showing proportions of women having had mammograms and Pap smears, and mortality rates and proportion diagnosed at local stage for cervical and breast cancer. In the spacefill representations, order is determined by column, color by row.

**Multivariate: Spacefill and map matrix**

Figure 4-33 shows a spacefill and map matrix. It has the same strengths and weakness relative to the map and scatterplot matrix that the spacefill and scatterplot do generally: the spacefill is better at showing nulls, potentially over-plotted observations, and relative
relationships, while the scatterplot is more immediately interpretable and allows the analyst to read values from locations in the plot.

One pair of variables that the spacefill is an aid in for the variables under analysis here is in the relationship between cervical mortality and breast mortality. The clusters of light values at the top and bottom of the spacefill inform the analyst of a bi-modal distribution.
Figure 4-34 shows a small multiples view combining map, scatterplot, and spacefill representations. As discussed above, the three forms can counterbalance each other. For example, in the Pap smear and mammogram combination that runs along the far left column in Figure 4-34, the scatterplot shows the numerical data values most clearly. It immediately shows how the values that have a “zero” value in the mammogram variable stand out. The map shows the spatial
pattern, with clusters of purple in Pennsylvania, representing a higher rate of mammogram screening, and more greenish values in West Virginia, representing a higher rate of Pap smear screening. The spacefill representation shows the seven null values that the two variables have in common most clearly. There is a relative concentration of lighter values towards the bottom of the representation, indicating positive correlation.

---

Figure 4-34: Small multiples plot showing proportions of women having had mammograms and Pap smears, and mortality rates and proportion diagnosed at local stage for cervical and breast cancer. The circled areas show the strengths of each representation type. The scatterplot shows the counties with very low rates of mammograms most clearly. The map shows spatial clusters of high and low values most clearly. The spacefill shows the correlation between null values, and the low-low correlation between Pap smear rates and mammogram rates.

---

**Coordinated Analysis Session**

The tools, when used together, are more powerful visualization devices than when used separately. Additionally, the advance that this research presents is not as much in any individual
component, as in the connected sum. To demonstrate the hypothesis generation potential of the kind of coordination strategies introduced in this research examples of such hypotheses are provided. Therefore, an integrated analysis session is presented here that demonstrates how the components may be used together, and shows some interesting hypotheses that are generated through the use of the interconnected tools. Because interactive, coordinated data analysis sessions are difficult to capture as textual descriptions, a video version of this analysis session has been created as is available at 

(http://www.geovista.psu.edu/members/hardisty/videos/coordinated_analysis.avi).

The data set is the same one as used in the comparative analysis, consisting of six variables. The first two variables, about screening rates, are proportion of women who have had Pap smears within the last three years, and the proportion of women who have had mammograms within the last two years. The next two variables are proportions of cancer that were diagnosed at local stage. The last two variables are age-adjusted mortality rates for cervical and breast cancers. These variables will sometimes be referred to in the following short forms: Mammograms, Pap Smears, Local Breast, Local Cervical, Breast Mortality, Cervical Mortality.

Figure 4-35 shows an initial view of the GeoViz Toolkit with the data loaded. At the start of this session, the PCP and small multiples matrix are visible. Other components are in minimized form along the bottom of the frame. The coordination is evident in one sense in that the different tools are showing the same data set. A DataSetEvent being broadcast by the Spatial Data Import and Transformation component achieved this, in combination with the data being received and processed by each individual component.
Figure 4-36 shows the result of a subspace event being coordinated across the components. The components were ordered by selecting the Pap smear variable in the Subspace Linkgraph, and then sending the subspace to all listening components. The Pap smear variable was chosen because it is a key screening indicator, cervical cancer being almost entirely preventable. One of the desirable outcomes of this analysis is some insight into how better screening can reduce the cancer burden in Appalachia.

The PCP and the small multiples display are two of those listening. Having the Subspace Linkgraph order the variables helps to show the slight positive relationships that we know exist in
the data. The slight positive correlations between pairs of variables are evident in the PCP, for example between mammogram rates and cervical cancer mortality. At this stage, however, it is difficult to see very many detailed patterns in the data, so we move to an automatic tour of different subsections of the data.

Figure 4-36: The GeoViz Toolkit, after selection and ordering of variables is performed using the Subspace Linkgraph

Figure 4-37 illustrates the appearance of several coordinated components during a selection animation “grand tour” of the data set. Grand tours are overall views of data sets, showing a set of specific views in sequence and giving the analyst a chance to find patterns that might be missed by manual examination (Asimov 1985). Grand Tours have proven effective in a geographic context (Monmonier 1989; Wilhelm, Wegman et al. 1999).
A striking result upon viewing a grand tour of this data set can be presented in static form (once found) by using selection to move over each variable in the dataset. Namely, the counties with the highest rates of cervical cancer (top tenth) are geographically clustered. None of these counties is in Pennsylvania, and many of them are near each other. Since cervical cancer is one of the key outcomes we are interested in, we would like to further understand this subset of the data.

Figure 4-37: The GeoViz Toolkit, showing the selection animator running. Counties with rates of cervical cancer mortality in the top tenth of this data set are highlighted.

Figure 4-38 shows how a bivariate coloring scheme is chosen that helps to illustrate which places are high in cervical cancer mortality, and low in proportion of cervical cancer cases diagnosed at local stage, or vice versa. This is accomplished by increasing the number of classes, then providing more contrast for the top end, by anchoring the second to last color swatches and
changing the green and purple hues at the ends to completely saturated colors. This color combination especially highlights those places that are high in cervical cancer mortality, but low in the proportion of cervical cancer cases diagnosed at local stage (in purple), and vice-versa (in green). This color scheme is broadcast to any interested listeners, and will serve to highlight cases that fall into these classes. In order to clearly examine the counties that have particularly high rates of cervical cancer mortality, we need to use another kind of tool, such as conditioning.

Figure 4-38: Bivariate classification in the map, highlighting counties with high rates of cervical cancer mortality, which are also low in proportions of cervical cancer diagnosed at local stage.

Figure 4-39 shows the result of the conditioning manager component broadcasting that the conditioning should be limited to those places that are highest in cervical cancer mortality. This conditioning is reflected in the map, in the PCP, and in the matrix components. One
observation that can be immediately made from the PCP is that the counties in this reduced data set seem to be more highly correlated than the data set as a whole, that is, the lines (representing counties) between the axes (representing variables) tend to slope upwards or downwards as a group; there are fewer crossing lines. This difference can also be seen in the scatterplot matrix, particularly in the case of the correlation between the Pap smear variable and the proportion of cases diagnosed at local stage, which has possibly interesting implications for improving screening, and lowering the cancer burden, in those counties.

Figure 4-39: Conditioning on cervical cancer mortality. Counties with high rates of cervical cancer mortality are shown, counties with middle or low rates are not shown. Geographic clustering can be seen in the map, with groups of counties in West Virginia and Kentucky. Attribute clustering can be seen in the PCP, with many of these counties showing similar multivariate characteristics.
Figure 4-40 shows how these insights can be followed up with using the spacefill visualization component. An interesting observation can be made here. There is one critical matrix element that shows the relationships between cervical cancer mortality (cervical cancer mortality is what the figure is conditioned on) and both breast cancer mortality (the bottommost variable) and cervical cancer diagnosis (the third from the bottom). First, the visible observations are grouped at the top and bottom of the element, showing that counties with high cervical cancer mortality tend to be either high or low in breast cancer mortality. Secondly, the observations are mostly grayish, showing that the counties with high cervical cancer mortality rates mostly have low rates of diagnosis at local stage. Finally, the few purplish counties are clustered at the bottom of the representation, showing that those counties with high cervical mortality and high screening rates tend to have low rates of breast cancer.
Discussion

This analysis has shown that potentially significant relationships between cervical and breast screening, mortality, and stage data can be extracted from using the GeoViz Toolkit that might not have been observable otherwise. Further, the coordination between the tools is of critical importance for making these observations possible. In particular, those places with particularly high rates of mortality from cervical cancer have some interesting commonalities. Firstly, they are geographically clustered. Secondly, they tend to exhibit a stronger positive relationship between Pap smear screening and proportion of cancers diagnosed at local stage.

Figure 4.40: The spacefill matrix after conditioning on cervical cancer mortality, showing only those counties with high mortality. Two interesting clusters of counties are highlighted.
when compared with the total data set. Thirdly, those places, if they also have higher than average rates of women having had pap smears, will tend to have lower rates of breast cancer. These exploratory observations will be followed up with confirmatory analysis.

**Conclusion**

This chapter has three main contributions to the design of geospatial demographic data visualization software, as well as contributions to the conceptual underpinnings for such software, to research on ESDA methods generally, and to broader InfoVis challenges. These contributions are categorized sets of events that support coordinated geographic visualization, an integrated toolkit using them called the GeoViz Toolkit, and a comparative analysis of bivariate and multivariate visualization tools with a data set that is of considerable interest in the research community. It is hoped that each of these could be leveraged by other researchers in related areas. Specifically, the contributions could be leveraged by re-implementing techniques developed here in other contexts, including in combination with a commercial GIS. In addition, the tools themselves, as freely available source code, could be extended in other directions. Beyond this, general principles for creating component-based, coordinated geovisualization software have been presented.

The first main goal identified at the beginning of this chapter and in Chapter 1 is to provide a description of the kinds of coordination that are desirable between components intended to support exploratory demographic data analysis. Having achieved this highlights the contribution that geographic visualization has to make for the larger questions relating to the design of geospatial demographic data visualization software. The abstraction, into events, of what should be communicated and acted upon between software components for classification
and symbolization of data, could be considered a practical system of geographic visualization, as well as a more general methodology for how coordinated geovisualization should be done.

Focusing on a more general level of contribution, this work has impact on: design of software, analytical practice, collaborative work, and the theoretical underpinnings of GIScience and InfoVis. For software design, this work contributes an architecture for geographic visualization that enables robust and extensible analysis environments. Although this work has focused on exploratory usages, this architecture supports coordination with confirmatory statistical techniques, which will be one of the next areas of research, especially leveraging ESDA techniques (Anselin and Getis 1992). For general analytical practice, if analysis is to freely move through different parts of hypothesis generation and conformation then disparate components must be coordinated, as no one monolithic piece of software does (or could) meet all analytic needs (Gahegan 1999). The work presented here is a step towards removing the barriers to effective geovisualization. This work also has implications for collaborative geovisualization. For collaborative geovisualization to become a reality, coordination among distributed components used by distributed people is critical. The ideas and mechanisms presented here provide one way forward. Indeed, this work advances the theoretical underpinnings of GIScience and InfoVis because most work in HCI related topics is still focused on interaction with self-contained applications. What is presented here is the beginnings of a system of geovisualization at the “between component” level, instead of remaining at the “within component” level.

This system is put into action in the GeoViz Toolkit. This toolkit provides a set of fifteen software components, created or extended as part of this dissertation research, which enable multivariate demographic data analysis and summarize and put into practice the concepts and tools described in the preceding chapters. This toolkit is freely available in source code form to interested researchers and software developers, and in binary, executable form to those interested in using it on their own geospatial data sets.
Finally, some of the components from the GeoViz Toolkit are put through a comparative analysis using data from the Appalachian Cancer Network (ACN), along with data provided by the National Cancer Institute. These data focus on cervical and breast cancer patterns, both spatial and statistical. These data consist of screening data, data on stage in the body at time of diagnosis, and age adjusted mortality counts for both cervical and breast cancer. Striking local clusters of cervical cancer mortality were identified, and differences in the interactions between screening and mortality were identified for counties with high and low mortality. This proof-of-concept analysis meets the third goal identified in Chapter 1.

The overall finding is that the tools need to be used together, as they have complementary capabilities. The findings are that the bivariate map tool, when used with complementary bivariate color scheme, can be effective for showing spatial patterns in the data, while the spacefill visualization tool can pick up some patterns that the scatterplot does not. From the multivariate comparative analysis, we find that here, too, each tool has its role. Scatterplots in matrixes are essential to enable easily interpreted representations of the data. Geographic maps provide visual access to spatial patterns, and can give the analyst effective visual access to large numbers of variables at once when placed in matrix form. The spacefill visualization component can identify some more subtle statistical patterns that the scatterplot or PCP may miss. The Linkgraph is the only tool that can summarize highly multivariate patterns in a unified, instead of pairwise, manner. The Subspace Linkgraph provides automatic variable ordering facilities that many of the multivariate visualization components used.
References Cited


Andrienko, G. and N. Andrienko (2001c). Interactive Cumulative Curves as a Tool for Exploratory Classification. 9th annual conference GIS Research in the UK, University of Glamorgan, Wales, University of Glamorgan: 439-442.


Crampton, J. (2002). "Interactivity Types in Geographic Visualization." Cartography and Geographic Information Science 29(2).


Chapter 5

Conclusion

The contributions of this dissertation research to the design of geographic visualization software for enumerated data, as well as to the concepts underlying such design, are described below. Then, some future research prospects opened by this dissertation are described. The contributions include the identification of important problems for geospatial demographic data visualization software at three levels (architecture, component, toolkit). Further contributions are made by introducing strategies for attacking these problems, and then by showing implemented versions of these strategies. The future research prospects opened by this dissertation include a variety of research questions, including ideas relating to: support of different uses’ ontologies, cross-platform interoperability, development of additional representation forms, and methods for preserving the state of an analysis session to support collaborative research.

Goal Assessment

The first chapter outlined groups of specific goals, each group addressing research challenges at a different level of design of geographic visualization software for enumerated data (software architecture, individual components, and toolkit). This section describes how this research has addressed these challenges and goals. Each section first restates the research challenge and the goals that address the challenge, and then comments on how well the goals were met, along with the factors that contributed to the goals being met, or prevented them from being met.
Software Architecture

The challenge identified at the software architecture level is to create a methodology for allowing software components to interoperate, while remaining open to change. I identified two goals at the software architecture level:

Goal 1.1: To find ways to leverage advances in component-based software, introspection, and reflective invocation to create more stable and robust geographic visualization software for enumerated data.

Goal 1.2: To enable flexible coordination between visualization components.

These two goals were met by introducing a strategy for linking geovisualization software components, and by providing an implementation of this strategy, called the Coordination Manager. This strategy uses introspection and reflective invocation on components, allowing components to be linked without explicit dependencies on each other. The overall strategy addresses Goal 1.1 by reducing dependency between classes. This reduced dependency increases robustness of software in two ways. Reduced dependency helps to reduce uncontrolled software failure because reduced dependency prevents failures in one part of a system from spreading to the rest. Reduced dependency also means that changes in one part of the software to better meet user requirements (or for any other reason) are less likely to require changes in other parts. This increased robustness against failure and robustness against change meet the first goal. The strategy addresses the second goal, of enabling flexible coordination between geographic visualization components, by using the same strategy. The Coordination Manager is able to automatically query components for what types of common visual and numerical aspects of data representation they support, and connect them on that basis.

Both were partially met. Goal 1.1 was generally well addressed by the component-oriented strategies used, relying on component introspection and reflective invocation. The advantages of components, reflection, and introspection described in Chapter 2 are real, and the
software developed in this research bears this out. The map component was the first, and core, component. Using component approaches meant that as pieces were developed – (for classification, and for color scheme application) the pieces could be independently reused in other contexts in the toolkit. Reflection and introspection were used in the coordinator component, which is a practical means of creating an integrated toolkit to which it is easy to add and drop components. On the other hand, these strategies do not offer a universal solution. If the design of the components does not allow for a particular piece of functionality, or if the components created are too large or too small, the flexibility of the overall system suffers.

Goal 1.2 was well addressed from the perspective of making coordination easy, but flexibility of coordination is limited. The coordinator provides a good default solution. It can connect the components in sensible ways. However, if the user wishes to use these events in a novel manner, the coordinator cannot help. For example, if the user wishes to express a brush in one component as a conditioning event in another component, the coordinator cannot accommodate them. Studio, by contrast allows the user to make customized connections, but at the price of a complex process of making those connections. Therefore, the coordinator and Studio are independent, but complementary, pieces of software.

**Individual Geovisualization Components**

The challenge at the level of individual geovisualization component design is to find effective means of interactively presenting highly multivariate data spaces to the analyst. Three main goals were introduced to meet this challenge.

Goal 2.1: To give analysts interactive, flexible control over data-to-color mappings in univariate and bivariate color spaces.
Goal 2.2: To develop methods that reduce the complexity of highly multivariate data spaces in a way that supports understanding of relationships among variables and effective choices of which variables to explore together.

Goal 2.3: To directly represent highly multivariate geospatial data spaces in ways that support common classification and symbolization choices.

These three goals were met by the development of strategies and implementations that draw from geovisualization, InfoVis, and EDA. These contributions were presented in two groups, those that allowed for direct geovisualization (representation of each observation and each relevant variable) and those that combined dimension reduction and node-and-link graphs to present data spaces to the user. The direct visualization approaches included a system of color anchors for data-to-color mappings, which were then used in a PCP and a space-filling visualization component. The solution for allowing direct interaction with a highly dimensional space was to use minimum spanning trees to reduce the data space, and present the results in an interactive visualization component.

All three goals for individual component development were partially met. Goal 2.1 was well met because a novel and potentially useful technique (color anchors) was introduced, but implementation needs to be improved to make this approach to color customization easier to use. Color anchors have considerable potential, and can be applied in a variety of situations. The examples provided in this dissertation all apply color anchors to a finite number of classes, but color anchors would work as well in N-class color assignments. Goal 2.2 was met, because the Linkgraph tool does enable exploration of highly dimensional data sets, but the implementation is limited. The combination of a summarizing tool like minimum spanning trees with a directly-manipulable graph visualization tools is a novel and powerful one, allowing the user to browse the local structure of highly dimensional data sets. However, the limitations on number of observations and number of dimensions are restrictive. Additionally, although the potential utility of the general concept of turning highly multivariate data spaces into graph structures that can be
browsed is validated by the implementation, minimum spanning trees do have some evident
limitations. One of these is that some local structure is lost, for example, a cluster of observations
that are near each other in the highly dimensional space might not all be shown as related in the
MST. Goal 2.3 was met in that the PCP and the space-filling visualization are tightly integrated
with the Visual Classifier, but the tools could be extended much further. Such extensions could
include adding more statistical information to the axes in the PCP, or adding more control over
the shapes inserted into the space-filling visualization component.

Coordinated Toolkit

The challenge at the coordinated toolkit level is to create an integrated geovisualization
toolkit that is effective at allowing analysts to discover previously unknown relationships in data.
Towards this end, three goals were introduced:

Goal 3.1: To build a typology of general categories for sets of visual and
numerical aspects of data representation that should be supported for coordinated
geovisualization.

Goal 3.2: To determine which specific visual and numerical aspects of
representation should populate these categories.

Goal 3.3: To give a proof-of-concept demonstration of the utility of the
coordinated toolkit by exposing interesting patterns in an epidemiological data
set via a coordinated analysis session.

These goals were met by introducing a set of categories (Selection, Dataset,
Symbolization, Classification, Spatial, and Temporal) for coordination between geovisualization
components. These categories were then populated by events that allow components to
communicate about geovisualization aspects from these categories. The components in the
GeoViz Toolkit were then introduced: these components use the events introduced earlier. A
comparative evaluation is performed of bivariate and multivariate geovisualization tools from the
toolkit, to evaluate how their strengths might be combined. Finally, I report on an example analysis session, which demonstrates the kinds of hypotheses that can be generated using the coordinated toolkit. This session identified a series of relationships in the cancer data set examined that will prompt subsequent hypothesis-based investigation. Among the relationships identified is a close geographic grouping of counties in West Virginia with high cervical cancer mortality, and a closer relationship between the screening rates and mortality rates in those counties than in the data set as a whole.

These goals were ambitious, so their being partially met can be considered a success. Goal 3.1 was met by a first attempt at delineating categories of coordination (an important problem in geovisualization), but the result suffers from being a first attempt. The grouping of data, display, and category seem sound, but this systemization will need further work. There are intermediate cases to consider, and each of these groups needs be expanded with logical sub-types to be more complete. Goal 3.2 has been met by providing a working implementation, but cannot entirely escape the limitations of the components to be coordinated. The coordination works. Selection in one tool is reflected in the others. However, the tools must be designed with coordination in mind, or they will not have the facilities to respond to the coordinated events. This often necessitates at least a partial re-engineering of the components (if pre-existing components are used – and the ability to use pre-existing components is one of the arguments for a component-based architecture in the first place). Finally, 3.3 is met because the example application does rely on coordination, but is not an entire success because of operations missing from the toolkit itself. The toolkit does not have many statistical routines built in, making the system useful for pattern discovery but not pattern conformation.
Future directions

In any framework and toolkit development effort, the author is left feeling like the tools and designs could be so much better if the author could just have more time to work on them. Further work designing geospatial demographic data visualization strategies will concentrate on extending both the design for geospatial demographic data visualization software and the instantiated tools that should result. The ways in which the current research should be extended can be organized along the same lines as the overall argument. This starts with abstract approaches that draw from component engineering, moving to more concrete designs for components to represent data in new ways (with ideas drawn from information visualization, exploratory data analysis, and elsewhere), and finally thinking about how these can all actually work together to solve real problems in geospatial demographic data, using approaches from geographic visualization. Each section will start with a description of the broader research questions that the techniques are intended to address, then explore what approaches might be fruitful, and then explore what further research questions might be addressed if the implementations goals are achieved.

Extending coordination to support user’s ontologies

The current coordination strategies are simple and powerful, but do not change on a per-user or per-task basis, making them a bit “one size fits all.” A challenging extension to the coordination methods (based on reflective method introspection and invocation) would be to automate the process of how the software works together to fit the users needs. This kind of extension would have substantial impact if achieved. A long-standing problem in the design of computer systems of all kinds is that the default model for computer usage is to have the
computer user change their working methods to fit the needs of the computer. To make real progress in both the adoption of existing visual analysis methods, and to develop truly innovative workflows with programs and analysts working in concert, the software needs to fit the user’s needs. This is a multi-faceted, long-term goal, and having geographic visualization software that can change its functionality to fit different categories of users is a step towards that goal.

One way of doing this would be to have a digital representation of the user’s conceptual model of their task, and to have different tool configurations fit different conceptual models (Gahegan 2003). Here, two ways of achieving this functionality are proposed. One would be to expand the metadata available for components and for events; the other is to use aspect-oriented programming to add additional support for conceptual models.

The first means, using metadata, could be approached from a few avenues itself. One approach would be to have additional data available on a per-components basis, so that a map component would, by default, give a view using enumerated units to an epidemiologist, and an image view to a remote sensing expert. These settings could be attached to analysis sessions, or fixed as part of a user’s profile. This would be equivalent to having user settings by category of user. This is similar to having an analysis “template” as in the approaches taken in Microsoft software. Templates are used by taking groups of settings like background color, font, etc. in PowerPoint and applying them to a whole presentation – either when it is constructed or after the fact. The difference lies in the fact that having settings available on a per-component basis would enable software customization for new applications as they are created, rather than having to create different sets of settings for each application.

Secondly, types of events could have user categories attached to them, so that selections would be automatically coordinated further for some classes of users than for others. A third approach would be to collect extensive field data about how actual analysts use these toolsets,
automatically capture it, and have the default configuration match these for different categories of users.

The second overall means of automatically variegating coordination would be to use aspect-oriented programming. Aspect-oriented programming is a new programming paradigm, like object-orientation or component-based programming. One of the best-known implementations of aspect-oriented programming is the AspectJ project, an open-source effort hosted by IBM. Aspect-oriented programming overall gives the ability to “step in” and perform operations without changing the underlying components. Adding logging and error handling without changing the underlying code are canonical examples of how aspect-oriented programming can be of aid, but coordination via introspection is another candidate. The appeal of aspect-oriented programming is that it would be adaptable to retrofit existing components without modifying them; and could work with other components not written with the aspect-oriented coordination method in mind. This contrasts with the metadata approaches listed above; using them would require specific code to be added to components to make them work.

If one or both of these methods can be realized, this will enable the beginnings of a new kind of geographic visualization. Both the individual tools and the overall toolkit can be incrementally modified to support specific kinds of visualization for different classes of users. This would also aid in the development of new kinds of functionality; the interface and functionality could be updated and tested for adventurous users (including developers), while maintaining older functionality for previous users. Some difficulties must be acknowledged about this overall program of research. Firstly, there is no class of user for which the needs have been thoroughly expressed and optimal interface designs found. Therefore, creating multiple kinds of functions for different users may just add layers of sub-optimal functionality. Secondly, creating and maintaining large software systems is already difficult; adding a complex set of additional requirements creates additional dangers that the system may crash under its own weight, or
become so inflexible that critical new functionality cannot be added. Thirdly, software changing its functionality can function as a negative as well as a positive. Microsoft Office has had adaptive menus added to their user interface, but not all users find these adaptive menus a help. Creating good documentation for adaptive interfaces is a challenge as well.

**Extending coordination across platforms**

The GeoViz Toolkit is written entirely in Java. There are currently no mechanisms to link this toolkit with other systems, such as non-Java database software, or GIS. The appropriate mechanisms for allowing interoperable spatial data handling and visualization are an active concern in the GIScience community (Voisard and Schweppe 1998). An extension to this toolkit would be to provide coordination across operating systems, and across the “Java Wall”. Java is often lauded as a cross-platform language; the riposte is that Java is not cross platform, Java *is* a platform. Interoperations between Java components are amazingly easy; interoperations between Java components and native binary components are cumbersome at best. This is a real barrier to the broad uptake of geospatial demographic data visualization software written in Java, including all of GeoVISTA Studio, and including the toolkit described in this report.

One possible way out of this impasse would be to extend the coordinator to accommodate key native libraries, such as MS COM (Component Object Model) or .NET components, which have similar event semantics and syntactic structures. This would only be practical for events that could be automatically translated from one model to the other. It would make the tools much more useful, however, and would allow the GeoViz toolkit components to coordinate with ArcMap components, to some extent. This would require writing a wrapper on the ArcMap side, matching its proprietary internal structures to open standards.
Addressing this problem of interoperability appropriately would open up research frontiers for this toolkit, and beyond. For this toolkit, providing interoperability would provide a platform for leveraging the spatial data handling capabilities of commercial GIS as a backend, and using commercial GIS as a destination for potentially interesting patterns to be exported to and further investigated. Beyond this, the general problem of interoperability is an open one. It is the responsibility of the research community to make sure that the large commercial solutions support open standards, to enable open GIScience.

One of my short-term plans, which will help the interoperability of the tools described, is to integrate much of the GeoTools2 (http://modules.geotools.org/index.html) data model into Studio. GeoTools2 has a number of capabilities that will enable software leveraging it to make progress on some of the future directions mentioned above. These capabilities include support for Open GIS Consortium standards, filters, and numerous types of spatial and non-spatial data.

**Generalizing visual classifier to other visual variables**

The original design of the Visual Classifier was not to make it a device for mapping data onto colors, but for mapping data onto arbitrary types of visual representations. Shapes, sizes, textures, and types of dynamic visual variables, could all be interestingly linked to data. There are research questions relating to this for both the implementation and the use of such visual variable capabilities. Implementation questions center around the appropriate code representation of these variables, and how they should best be related to data. There is a difficult balance to be found between abstraction and speed. Abstraction can be a powerful tool to make the code more modular and extensible; but it can also make operations harder to write and slower to execute. Java provides a good set of graphical primitives, so the challenge becomes one of abstracting what is particular about these graphical objects from what could be applied in other contexts. For
example, creating routines that can appropriately scale and transform shapes, such as a fisheye lens, can be applied in many different visualization components, like maps, scatterplots, and parallel coordinate plots (Fekete 2003).

The uses of multiple visual variables are manifold. Two of the most important are that of different means of expression, and that of composing variables to express more variables simultaneously. Firstly, different visual variables may be better at expressing different types of information or relationships. Secondly, using visual variables like symbol size, symbol color hue and symbol color value simultaneously, in a system that allows users to manipulate data-to-display mapping interactions may result in a representation which can make multivariate relationships evident that would otherwise not be. This is an open question, in part because there is a lack of good geographic visualization platforms on which to test such ideas. A compact visual representation like the visual classifier would aid considerably in giving the user an easy route into controlling the data-to-visual appearance mappings.

**Integrate Treemap representation**

The Treemap representation alluded to in Chapter 3 as a kind of space-filling visualization could be profitably integrated into the rest of the GeoViz Toolkit. The Treemap (and other mosaic-types plots) excel at combining categorical with numeric data (Friendly 2002). One obvious application in Studio for this would be to use the Treemap to map hierarchical geographic data, like tracts, counties, and states, an application that has already seen use elsewhere. Another research question would be to determine the most effective uses of Treemaps for showing different categories, whether user derived or machine learning derived. In either application, treemaps would fit well inside matrix-type visualizations such as the bivariate matrix or the small multiples matrix. Among the possible mappings are using bivariate color schemes, as
used in the bivariate maps, or using color and size, color and order, or size and order. Allowing the user to interactively modify the categories would help investigate the area of human-computer collaboration on classification and categorization of data.

Another interesting extension to the stable of matrix-ready representations is a PCP. PCP representations with three axes, with the same middle axis appearing in all matrix cells, and the two outer axes varying according to the position of the PCP in the matrix, would provide unique information for each cell. This would require a computationally efficient rendering canvas, in order to render many of them at once.

**Creating Python scripts for configuration**

A related idea for customization is to provide a Python interpreter inside the GeoViz Toolkit, and inside Studio. A limited proof-of-concept implementation has already been achieved (Maegill 2003). The type of Python interpreter is a Java one; the Java implementation of Python goes by the name of Jython. Here, references to Python will be assumed to be to Jython; which is not capricious because Jython is Python. Most Python programs written with the more-popular C implementation of Python will run without modification on a Jython interpreter.

One advantage of having a Python interpreter inside a Java environment is that Java is a static language, for good and ill. The static constructs are like contracts between the programmer and the compiler, with the compiler always informing the programmer when she makes mistakes. Static languages force the programmer into a code – compile – run paradigm. This encourages good programming practices, but discourages play and fun in programming, which can be important creative aspects of programming. Perhaps more importantly, end users could build their own custom formulas, data transformation procedures, and other custom functions without any understanding of the underlying Java component architecture. Other advantages of scripting
languages, such as compact code representations, and easy customization, would also be obtained.

**Saving preferences**

Since one of the primary arguments for coordinating classification and symbolization choices is to propagate the choices of the user in one component to other representation forms, the fact that the current software does not preserve user preferences across sessions is disappointing. Additionally, the prospect of being able to save and transport interesting program states in the analysis of geographic data would enable better sharing of and collaboration on geographic analysis (Gahegan, Wachowicz et al. 2001). Both automatic and manual approaches to saving user preferences would be valuable additions. Automatic preferences would save classifications and other settings on a per component basis. This could be automatically saved across sessions with no effort on the users part (but a considerable effort on the programmers part) by utilizing the Java preferences package, introduced into Java 1.4.

The manual approach would be to save the status of the beans to disk upon request, and be able to retrieve it when needed. One current mechanism for doing this kind of preference saving would be to have the state of the components serialize, or save, along with the beans, when they are saved as part of a *Studio* design. A *studio* design is the sum total of the beans and their connections. The current serialization mechanism in Java is that provided by the Java Bean standard. This states that all bean properties that are not marked “transient” should be serialized using “get” and “set” methods that match the name of the property. Further, the serialization can be customized by providing such information in the Bean’s associated BeanInfo class.

Another, complementary, and perhaps more widely applicable approach would be to concentrate on serializing the coordinated events, and saving the events, with the associated state
data, as opposed to saving the entire contents of the components. Thus, a dataset event would be saved, as well as current classification and color choices. One advantage of this approach would be that the resulting saved session info could be applied to any set of coordinated components, either inside or outside of Studio proper. Therefore, one could save different sets of session information, and have these applied to different designs.

This “serializing events” approach would have a few limitations. One is that large geographic data sets are not practical to save using the standard XML serialization mechanisms. This is because each datum about each geographic entity, whether it is a geographic point along a boundary or an associated data value, is saved as a text entry, along with tags identifying them. This causes massive inefficiencies in internal handling and saving of geographic data in this form. A test using a data set of U.S. counties with 10 variables for each county resulted in an XML document with more than one million lines. This is not practical to handle with today’s computing resources. Therefore, a custom delegate should be developed for the data set events. Additionally, the only session data that would be saved are those data that are open to coordination, and are actually connected to the coordinator. Another limitation is that some events do not store the needed data directly, but maintain a reference to the object doing the work. If this object is not present, or if this object does not serialize correctly, the overall design will fail to serialize correctly.

Enabling saving and transporting of “analysis states” leads to some important research prospects, among them improving collaboration at a distance, and allowing the analyst to better preserve the analysis process, rather than just the end results. Collaboration at a distance, using geographic visualization software, could be considerably enhanced if analysts were able to share their analysis environments at any time. Similarly, if the analyst is able to preserve the state of the visualization software at a moment of inspiration, this help to enable analysis sessions to be more focused on the process of science, rather than just the results.
Final thoughts

Many research frontiers in geovisualization, from geographic representation components, to geo-collaboration, will need to consider the possible advantages that can be reaped by using a component-orient approach. The creation of the best possible geovisualization methods demands that researchers pool their ideas, and build on each other’s efforts. Since no one, over-arching strategy can serve the multitudes of needs met by different geovisualization techniques, the only way forward is to find methods for allowing disparate approaches to work together. This is also a possible means for making sure contributions in our discipline can serve a wider public, by allowing interoperation with widely-used commercial GIS packages. For geo-collaboration to meet its potential, especially in the case of analysts working simultaneously at a distance, careful coordination of different analysts’ geovisualization environments is required. This challenge can best be met by strategies such as the ones introduced in this research.
References Cited

http://www.lri.fr/~fekete/InfovisToolkit/


