In M. Meybury (ed.), Intelligent Multimedia Information Retrieval. AAAI/MIT, 1997, 83-111.

Sketching, Searching, and Customizing Visualizations: a Content-based Approach to Design Retrieval

Mei C. Chuah, Steven F. Roth, Stephan Kerpedjiev

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
Tel: +1-412-268-2145
E-mail: {mei+, steven.roth, kerpedjiev}@cs.cmu.edu

http://www.cs.cmu.edu/~sage

ABSTRACT

We present new techniques for retrieval of data-graphics and a system, SageBook, that employs these techniques to facilitate the process of visualization design. Design is an important activity in many different disciplines, including engineering, science, and business, but current systems provide little support for non-expert users to design new graphics for use in data analysis. SageBook's approach is to provide expertise through the retrieval and reuse of previously successful designs. The design task places new demands on retrieval technology because it requires not only a good search engine but also effective tools to pose queries, browse results, and adapt previous designs for reuse. Despite our focus on data-graphic design, the concepts presented can be transferred to other design activities.

1. INTRODUCTION

This chapter will discuss retrieval as it relates to the problem of graphic design, an important activity in many disciplines and tasks. Graphics are used by analysts in many domains to analyze trends, detect patterns and anomalies, and answer focused questions. Data analyses are also performed by statisticians to identify relationships or detect problem areas. These activities are classified as exploratory data analysis (Tukey 1977). In addition to analysis,

graphics are useful for succinctly and clearly communicating information to others in presentations. Whether for analysis or presentation the success of these tasks depends on the ability of people to design effective graphics of their data quickly.

In the area of visualization design, the last decade saw significant progress in developing intelligent tools that help users construct data-graphics (e.g. Mackinley 1986; Roth and Mattis 1990; Casner 1991). A common feature of these systems is their ability to generate graphics that integrate multidimensional data. Most commercial applications, including popular spreadsheet systems, produce only simple graphics that tend to isolate data attributes in separate charts. In contrast, these intelligent systems integrate multiple attributes into one graphic using a variety of composition techniques: multiparameter graphical objects, space alignment, and grapheme clustering.

Another major problem that has been addressed only recently in the SAGE research is providing users with an interface that helps them design custom visualizations (Roth et al. 1994). This research stresses the need for a combination of user-controlled interactive design tools and automatic design mechanisms. Their assumption is that design is inherently a dual process of constructing or assembling graphic elements into composites in a bottom-up fashion, as well as a process of considering previous examples that might be relevant to current needs. Thus, these processes suggest complementary tools for specifying graphics constructively and browsing previously created graphics to reapply them.

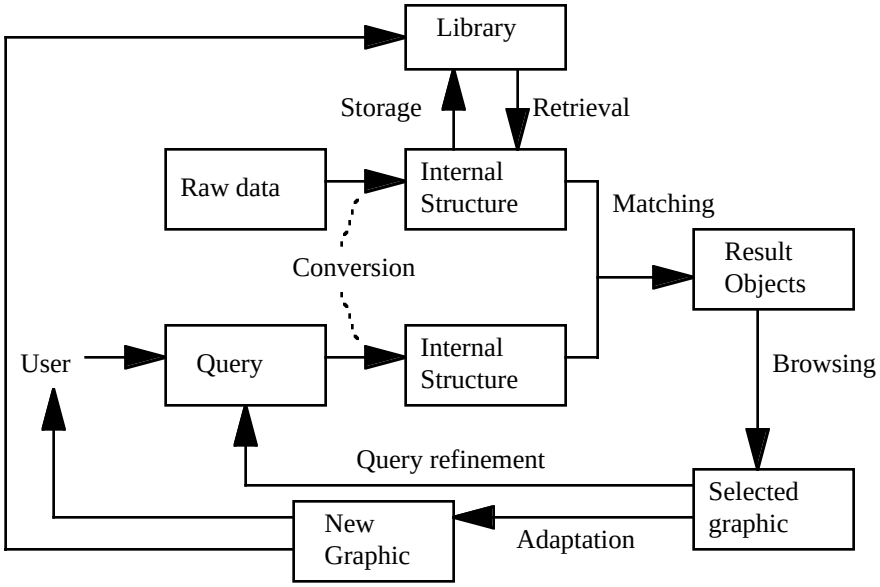

Figure 1: The retrieval and reuse process for designs

To support these processes, two tools were created. SageBrush is a tool with which users sketch their design ideas; the intelligent design engine of SAGE then converts this sketch into a data-graphic. Moreover, the sketch may be incomplete, in which case SAGE attempts to complete the graphics by selecting and composing additional graphical elements and properties. Although the interactive graphic design techniques supported by SageBrush are useful when people know the graphic they wish to create, it is still necessary to provide them with alternative graphics when they are unsure how to visualize data and need to browse through design alternatives. To provide this type of design support, we developed another tool called SageBook. It is closely integrated within the SAGE system and provides retrieval capabilities that help users extract relevant visualization designs and adapt them to their needs. The major processes occurring in SageBook are shown in Figure 1.

Central in the retrieval and reuse process shown in Figure 1 is a library (store) of visualization designs. This storage infrastructure is expressive in such a way that it accommodates important design elements and supports easy conversion of existing designs into its internal storage language. In Figure 1, the rectangles labeled "internal structure" represent designs or queries in the internal language. Users can query the stored designs

based on their current tasks and data. Query interfaces help users communicate to the system the types of designs that are desired. Therefore, the interface should closely match the users' mental model of design elements in the intended domain. Before matching a query with library entries, the system converts that query to the system's internal storage language in a way similar to the conversion of the original designs.

The internal description of the query is then used to retrieve library entries. There are two types of retrieval algorithms based on *exact* and *similarity* matching. Exact matching returns designs that fulfill all the criteria specified in the query, while similarity matches are less stringent and may return designs that contain enough, but not all, of the query elements. In design, one important use of graphic libraries is for getting new ideas from past successes. Since exact matching would severely limit the number and types of designs retrieved, it is critical to be able to retrieve designs based on some similarity measure. Similarity matching, however, can result in a large number of hits, especially when the library is well-populated. In order to effectively process the search results, users need tools that can help them organize the retrieved graphics. For example, users should be able to quickly browse through the retrieved graphics, collect them across multiple search sessions, and group them according to their importance, the data stored, or the graphical elements used. Finally, after selecting some designs from the library, users need to adapt these objects for current use. This might involve integration, addition and deletion of elements, altering the attributes used, or even completely redesigning the graphic.

In summary, there are five components of the retrieval and reuse process: query, storage, search, browse, and adaptation. These components are all important to understanding the difficulties involved in supporting the retrieval and reuse process in the design domain:

Need for integration with other activities. The retrieval activity is usually not performed in isolation. In design domains, users retrieve information to get ideas or for integration into new designs. These tasks require not only the retrieval of objects but also the manipulation of those objects. To be a useful appliance for a designer, the retrieval process must occur seamlessly coupled with other tools. They need to be able to search libraries in the midst of other activities and make use of retrieved artifacts in their design workspaces. This includes

tasks of querying, retrieving, browsing, and integrating. Most current retrieval systems only provide support for a very narrow part of these processes.

Designs are complex artifacts. Users' expectations from a retrieval system vary with the domains and tasks of interest. As it has been pointed out in (Griffioen, this volume), the retrieval process is domain-dependent because users need to search based on semantic content or embedded information. Hence, in order to effectively support the retrieval of designs, it is crucial to identify and represent the critical elements of a design within the system. Because designs are usually complex, it is difficult to capture all the properties of interest to the users.

Support for non-expert users. Most retrieval systems assume a certain level of user expertise and provide little support for users who are not experts or have little computer experience. In the case of data-graphic design, many of the users are analysts, planners and decision makers who lack design knowledge. In addition, design tasks often require a fair amount of computer experience because users need to manipulate the objects retrieved. To solve these problems, intelligent support should be provided in the form of design assistance or critiquing.

The communication problem. It is often difficult for users to convey their intentions to the system. This is especially true when specifying spatial or temporal structure. In order to improve the usability of retrieval systems, users must be able to communicate easily with the system, and support should be provided for articulating and refining queries and for understanding search results. Support should also be provided to allow smooth transitions between the different activities associated with the retrieval and reuse process in design.

In this paper we present a content-based[1] retrieval system, SageBook, that addresses all of the above issues in the domain of data-graphic design. Although our work focuses on data-graphic retrieval, the concepts and techniques developed, as well as the tasks supported, can be generalized and applied to other design domains. SageBook provides the following functionalities to support graphic design:

1By the content of data-graphics, we mean their graphical properties (graphical objects, properties and relations), how they encode data attributes (i.e. their mapping to data) and the abstract characteristics of data relevant to design. We are not refering to the meaning or interpretation of the data values contained in graphics.

Visual and context sensitive workspace for retrieving data: SageBook provides a visual workspace for storing and managing sets of data. Each data set incorporated within a graphic becomes a first-class object, retrievable based on cues relevant to users, such as visual appearance and data characteristics. Once retrieved, these objects can be browsed or interactively organized into groups.

A library of visualizations: When users are unsure about how to design a data-graphic, they can use SageBook as a library of examples, both for ways to visualize their data and to learn the design capabilities of the system. SageBook can be searched or browsed for prior data-graphics that have particular data or graphical characteristics. For example, one can retrieve all charts that have networks embedded in them to see how the lines and nodes can be embellished with additional graphical objects.

A tool for rapidly considering alternative graphic designs: Data-graphics can be created through a constructive process of selecting and arranging graphical elements (Roth et al. 1994). However, even when users are skillful graphic designers or use an automatic presentation system (e.g., APT (Mackinley 1986), SAGE (Roth et al. 1994), or BOZ (Casner 1991)), it can be very time consuming to generate many different data graphics that express the same data set in order to choose the most effective one. Even expert designers often need ideas when working with new data sets, perhaps ideas accumulated as a result of other users' successful attempts to visualize similar data. SageBook can quickly display a large number of browsable graphics, all related to a user's data set. Of course, this assumes that a portfolio of graphics has been accumulated over a sufficiently large range of data-types and graphic styles to provide a variety.

A tool for customization of prior designs: After a set of data-graphics is retrieved, users may request that one of them be used to create an analogous graphic for their new data (i.e., reuse the design of the graphic for a new data set). Users may also modify the designs from retrieved graphics. Thus even though a prior graphic design may not exactly match a user's goals, the parts of the design that do can be reused; those parts that do not match can be removed or altered. SageBook allows users to combine graphic design elements from several previously created data-graphics.

Current data-graphic design tools, particularly those provided with spreadsheets, do not support design retrieval. As a result, previous designs can only be retrieved by memorizing file names or exhaustively looking through all the data-graphic files. SageBook addresses this issue by providing users with support for storing data-graphics, formulating queries, retrieving, browsing and adapting data-graphics to suit current tasks. In the following sections, we will discuss how each of the five sub-tasks in the design retrieval process are supported by SageBook.

2. QUERY INTERFACE

Users must be able to easily communicate their search requests to the system. Effective query interfaces will have a direct mapping from the user's model of the objects to be retrieved to the query expressions in the interface model. Current query interfaces can be divided into five categories: command language, direct manipulation, keyword, query by example, and sketch.

To use a command language (Chang, Lee and Dow 1992; Rabitti and Savino 1992), users are required to learn the primitives and the syntax of the query language. Such languages are usually robust but difficult to learn and use.

Direct manipulation queries are created by manipulating widgets, menus, and objects. These interfaces are easy to learn but are less robust than command languages. This is because users can only make queries that have already been predefined. There is not much opportunity for formulating the complex queries that are possible with a command query language. Recently, Papantonakis and King (1995) developed Gql, a visual language whose expressive power is comparable to that of SQL. They also reported that a small-scale experiment had shown time decrease for formulating queries compared to text input. However, this type of language still requires that users develop a complete mental model of a language of the complexity of SQL, which increases learning time. An alternative approach (Young and Shneiderman 1993) exploits the metaphor of water flowing through filters for creating Boolean queries. An experiment has shown that there is a significant difference in the total number of correct queries favoring the Filter/Flow approach to text only SQL interfaces.

Keyword queries have long been used in retrieval systems. Keywords have the advantage that they require no learning time. Users simply enter a sequence of words and the system does the matching based on those words. However, keywords suffer from not being able to convey more complex relationships (e.g. spatial relationships among objects). In addition there may be mismatches between user-generated keywords and system keywords (Borgman et al. 1988). Many current systems complement image analysis with keyword matching to increase precision and recall (Kato 1992; Smith and Chang, this volume).

Query by example (Kato 1992; Holt and Hartwick 1994) is a powerful query method. It has very low learning time because users simply have to select an example object that represents what is required and submit it as the query. Employing this method, users can convey complex queries because the example object submitted is capable of representing as much semantic and syntactic information as any other object in the database. However, this method may be problematic when users cannot find an example image that is a good representation of what is desired. In such cases, users may have to look exhaustively through the library to find a suitable example. For this reason, this method is typically used in query refinement.

Sketch queries are most common for retrieving images. There are two types of sketch queries: *free-hand sketch* (Holt and Hartwick 1994; Nishiyama 1994) and *object manipulation sketch*. In free-hand sketching, users freely draw the query using a mouse, pen or other input device. There is no limitation on the set of permissible object types. The sketch is then analyzed, and important features are extracted (e.g., spatial relationships and shapes) and used for matching. Even though users may freely sketch many different types of objects, the system will only be able to understand the forms that are representable within its internal language. In object manipulation sketching interfaces, users construct sketches from available primitive objects, usually arranged in palettes. Although this method seems to be more limiting than free-hand, the fact that it does not need image analysis makes it much more efficient. In addition, users are guaranteed that all elements in the sketch will be fully understood by the system and that there won't be any error in interpretation. Object manipulation sketch interfaces are usually appropriate for systems that address focused domains.

2.1 SageBook Query Interface

SageBook allows users to query the system based on graphical properties and/or data properties of the stored images. Users form queries through an object manipulation interface called SageBrush (Figure 1). It provides a palette of spaces and graphemes as well as a view of the data, from which users create sketches of graphical elements or select subsets of data attributes. Sketches are constructed by simple drag-and-drop operations. For example, to create the sketch in Figure 1, the user dragged a chart space from the top palette to the working area, dragged a line, a mark, and a text grapheme from the left palette to the area inside the chart, and "opened" the line grapheme by clicking on it, so that all graphical properties pertinent to lines get visualized as icons (in the case of lines, four positional properties, color and line thickness are relevant). The user might have further specified this sketch by dragging data attributes to these property icons. Any data attribute mapped to a graphical property gets interpreted as a directive for encoding that attribute by that property. In this case, the *average temperature* data attribute has been assigned to the color of the line. The sketch and the set of attributes at the bottom represent a query, which SageBook matches with the entries in the design library, returning the graphics that fulfill the graphical and/or data constraints specified by the user. Query interface details are provided in (Chuah et al. 1995).

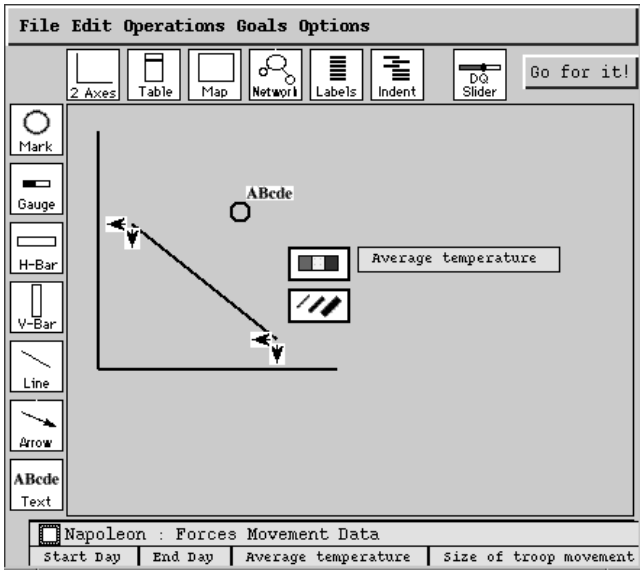

Fig. 2. SageBrush: The SageBook query interface. The space and grapheme palettes are located at the top and to the left of the interface, the data area is at the bottom, and the sketch is constructed in the middle of the interface.

Unlike command language queries, users do not need to know a complex vocabulary for describing content. Instead of learning the terms that the system uses internally to refer to axes, map spaces, interval bars, gauges, indented text, etc., SageBrush enables users to select and arrange spaces (e.g., charts and tables), the objects contained within those spaces (e.g., marks and bars), and the objects' properties (e.g., color, size, shape and position). Likewise, users do not have to learn the terms for describing the characteristics of data, such as scale of measurement (nominal, ordinal, and quantitative), or the relationships among data attributes (functional dependency, interval, and 2D coordinate). Instead, they just select some or all the attributes listed in the SageBrush data area. In addition to serving as a query interface, SageBrush can also be used to construct data-graphics and to manually adapt retrieved data-graphics. Because of SageBrush's multiple functionality, any data-graphic that can be constructed can also be queried.

SageBook uses object manipulation sketching because it supports a focused domain, in which all the primitive objects can be identified and made available to the user. An object manipulation sketching interface is more accurate, simpler and requires less processing. In

addition to the object manipulation interface, SageBook supports query by example. Users can query the graphic library either by giving a graphic example, a data example, or both. Since we are complementing this method with a sketching interface, users can just sketch what they want when no examples are available.

Once a query has been formed and submitted, it is translated into a language (design-directives) that is understood by the system components. When users select a data set or construct a graphical query, the system extracts the characteristics of each selected data attribute and graphical element and then reformulates the query in terms of underlying data and graphical properties.

SageBook does require data objects to be characterized when the data is first created. This characterization must be provided by database creators (e.g., in the form of relation schemes), by analysis modules that examine data values, or by modules that acquire this information from users interactively. However, once data is characterized and stored, users need not be aware of the characteristics or the language that is used to describe them.

3. STORAGE

The storage of objects has two main components: *storage vocabulary* and *method of translation*. Storage vocabulary refers to how objects are represented and determines which criteria can be used for matching. In principle, graphics could be stored in raw format (e.g., a bitmap, a video sequence, raw text) or in a formal language. The advantage of using a language is that less storage space is required and the search is more efficient. In addition, language representations have richer expressiveness and explicitly show relationships and structures that are implicit in images. Finally, language representations are easier for the system to manipulate. Typically, the disadvantages of using formal representations are that information may be lost in the translation from raw format to formal language and that the translation may introduce errors. The types of languages used in current systems include: low-level object attributes (Row and Frew, this volume; Pentland, Picard and Sclaroff 1994), entity-attribute-relation language (Hibler et al. 1992), and object-oriented language (Griffioen, Yavatkar and Adams, this volume; Halin and Mouaddib 1992).

Method of translation refers to how the raw formats are translated into the internal language of the system. Objects that are stored in raw form do not need a method of translation. Current methods include: object analysis, spatial and temporal parsing, model descriptions, and manual translation. Object analysis involves generating object properties and structure by using low level processing routines. For example, histogram distributions and segmentation fall under this category. Object analysis techniques (Row and Frew, this volume; Pentland, Picard and Sclaroff 1994) are general (i.e., domain independent) but are not able to produce higher level semantic content such as object relationships and other embedded information. Temporal and spatial parsing (Lakin 1986) retrieve object structure and relationships by using the location and temporal occurrence of those objects. Such techniques produce more structure and semantics than the previous method but require domain knowledge. Model descriptions refer to systems that have access to the full object descriptions. This information is usually not available unless the search objects were created electronically as well-defined models. Some examples include computer modeled scenes and CAD models. Finally, manual translation requires a person to annotate the search objects with relevant descriptions. This method is general and can produce complex semantic information; however, it requires a significant amount of human effort and may be inconsistent. Automatic indexing, when possible, is preferable to manual methods because the object library can be easily populated, maintained and organized for efficient search.

Yet another element of storage is its *organization*, which is extremely important for very large document bases. A poor organization may make the search time prohibitively high. Frequently used instruments for storage organization are indexing, hashing, and clustering. Indexing ensures direct access to the objects that have a given value specified in the query. Hashing organizes the documents according to the values of a hash function which may combine the values of different attributes. Clustering puts similar objects in continuous areas. Depending on the type of objects and methods of retrieval, one method of organization may be preferable to others. SageBook uses hashing and indexing techniques.

3.1 SageBook Storage Vocabulary

SageBook uses an expressive object-oriented vocabulary for describing the graphical and data relationships contained in data-graphics so they can be searched by content. The objects represented in the language are reflected in the query interface. This enables SageBook to translate user requests accurately and unambiguously (i.e. without vocabulary mismatches). Preliminary user tests show that users can construct queries with very little learning time, which indicates that the query interface and the vocabulary model correspond well with the user expectations.

A common data and graphic representation is used by all the modules of our system. This vocabulary is capable of expressing the syntax and semantics of data-graphic designs, and characterizing the data contained within them. It is able to express the spatial relationships among graphical objects, the characteristics and relationships among data-attributes. Most other retrieval systems do not provide such a detailed and expressive internal language.

3.1.1 Data Representation

The data characterization language expresses scales of measurement (nominal, quantitative, ordinal), structural relationships among data (e.g., between the endpoints of ranges and between the two attributes of a geographic 2D coordinate), and dependencies among attributes (e.g., whether each person has one or more birthdates, residences, children). We describe some examples in Section 4.2.2. A detailed description of the data characterization used in SageBook can be found in (Roth and Mattis 1990).

3.1.2 Graphic Representation

The internal representation describes each data-graphic as a design-specification, which contains one or more related *spaces.* Spaces can be related by being aligned, either vertically or horizontally. Spaces can only be aligned when the data types represented by the common axis are consistent. For example, *cost-of-labor* and *cost-of-materials* are two consistent attributes because both express dollar amounts. However, *person-weight* and *person-height* are not consistent attributes because weight is expressed in pounds while height is expressed in inches. In Figure 3, which shows a data-graphic from Napoleon's 1812 march on Russia, the

spaces are vertically aligned by the common data type *date*. The x-positions of the lines and marks in the upper chart and the bars in the lower chart all encode dates and therefore can be portrayed with a common axis.

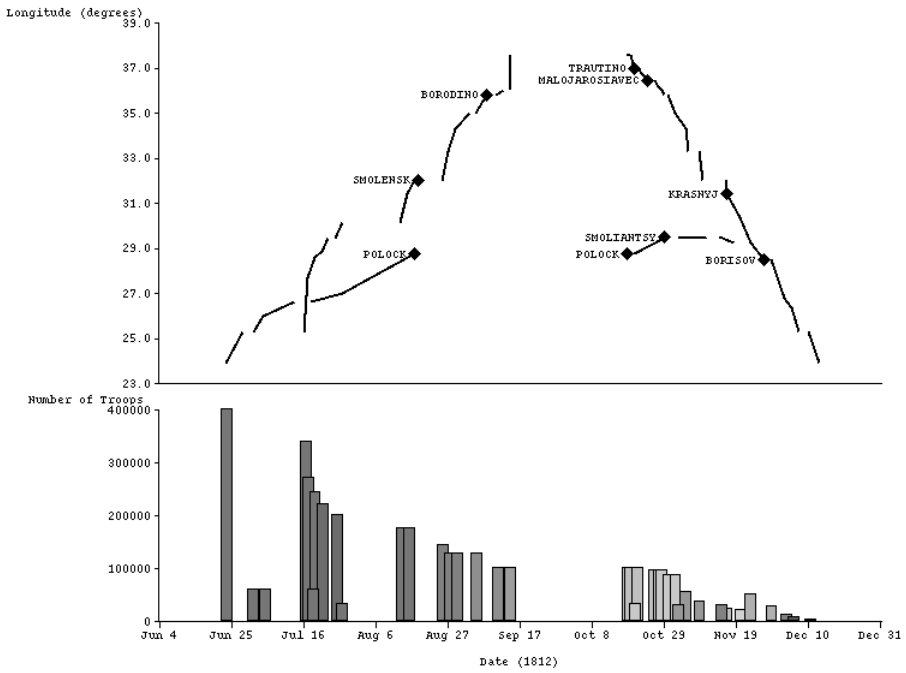

Figure 3: A two-space SAGE graphic

Each space represents groupings of graphical elements that are positioned according to a single layout discipline. There are many types of layout disciplines; map, chart, table, and network are the most common ones. The graphical elements contained within a space are *graphemes*. Examples of graphemes are marks, bars, text, lines, and gauges. Each grapheme uses different properties to encode data. Some of these attributes may be used to encode attributes, to distinguish different relations shown in the same space, or to convey a relation to another grapheme. For example, the data graphic in Figure 3 presents spatial, temporal and quantitative data. The positions of the labels are relative to the positions of the marks, which encode the *east-west-location* and *date-of-battle* attributes. Graphical attributes not encoding attributes or relations have default values (e.g., the size and the shape of the marks).

Figure 4 expresses the data-graphic in Figure 3 in terms of its constituents. The data-graphic contains two vertically aligned spaces, both of whose layout discipline is chart. Within the first space there are three sets of graphemes : a line, mark and label. The line position, the mark position, and the text lettering all encode data values, while the text position is relative to the position of the mark. The second space contains a set of bar graphemes using x-position, length and color to encode data.

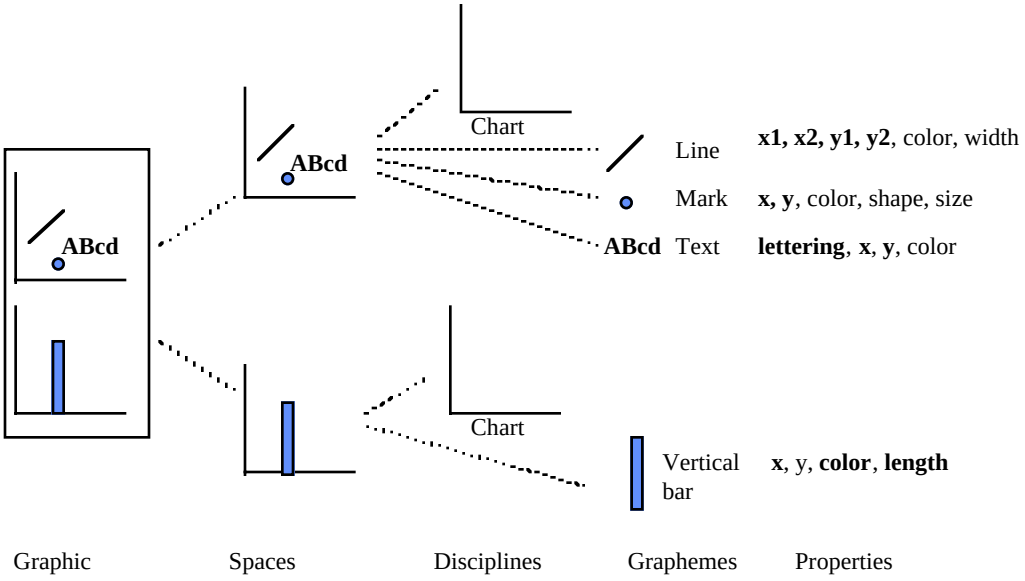

Figure 4: Constituents of the data-graphics in Figure 3 (the graphical properties that encode data attributes or are relative to others are printed in bold)

Thus, the *content description* language specifies classes of objects, the spatial relations among spaces and among graphemes, the graphical properties of objects, and the way they are assigned to data.

3.2 SageBook Method of Translation

The data-graphic descriptions used by SageBook get generated when the data-graphics are first created by SAGE, an automatic presentation system (Roth and Mattis 1991). When a SAGE data-graphic is saved, a description of it is stored in conjunction with an image reduced in size for rapid viewing (a large image is generated as requested from the description). This description is later used by SageBook during the search process. Because the same description

is used to generate the data-graphic, the stored representation contains at least as much information as the graphic. This is not true in other retrieval systems where the stored representation contains only a small subset of the object information and inconsistencies may exist between the stored representation and the object itself. The disadvantage of SageBook, however, is that only graphics created by the SAGE system can be stored. It is possible that, by using spatial parsing techniques (Lakin 1986) and domain knowledge, we can analyze general data-graphics and encode them into our language. Information on the graphical elements (e.g. spatial relationships) can be derived from visual processing, and other information such as data properties and relationships can be obtained from the database. Note that this is not a big change to the current system as it only affects the conversion methods from raw images to the system internal structure (cf. Figure 1), which is a small part of the system.

4. SEARCH

Search is the process of matching stored objects with user queries. There are three main issues in search: the match criteria, the type of search and similarity model, and finally, matching objects that have multiple types of data. The match criteria used in search depend on the storage vocabulary (presented in the previous section). Depending on the expressiveness of that language, the match can be simply based on keywords or might instead match object structure, relationships, and attributes. The richness of vocabulary in SageBook allows the matching algorithm to not only retrieve objects based on their properties but also on object structure and relationships among objects.

There are two classes of search - exact search and similarity search. Exact search will return only those objects that exactly match all of the constraints specified in the query whereas similarity search will return a set of objects that meet some similarity metric but may not conform to all the specifications of the query. Garber and Grunes (1992) showed that similarity matching is important and useful. The problem with it however is coming up with a good metric for judging similarity. How different attributes of the stored objects should be weighed in the similarity algorithm is based on a user's perception of which attributes, relationships and elements are more important. Current methods for determining similarity

include neural networks, case-based reasoning and other uses of domain knowledge. Both neural nets and case-based reasoning require users to train the system on a set of examples. In fact Row and Frew (this volume) show that even though results are promising, they are still inaccurate and require a significant amount of training. In SageBook, the different similarity matching algorithms are based on specific knowledge of graphic design.

Another issue in search algorithms is how to deal with objects that contain data of different types. For example, in video there is an image stream, a text stream, and an audio stream. To effectively search for such objects, we need to not only process each data type individually but also combine their results. Combining the results from processing each data type improves search accuracy and enables users to convey constraints based on more than one type of data. Mirialdo and Dubois (this volume) presented a method for combining results of various data types through the use of agents. Similarly, in SageBook, support is provided for users to search based on either graphical elements or data content.

4.1 SageBook Match Criteria

The search strategies in SageBook are based on the structural properties of the graphical and data elements in a data-graphic. The storage vocabulary, enables matching on semantic content and relationships among objects. SageBook's match criteria recognize positional relationships (including clusters, alignment and containment), graphical properties (e.g. color, shape, size), and data mappings (i.e. the assignment of graphical properties to data attributes). The search algorithm is able to make fine distinctions in graphics, for example the differences between Figures 5 and 6. Figure 5 shows a data-graphic of project-management data. The interval bars indicate the time spans of activities of different types. Their y-position and color properties encode the *work-type* and *status* attributes, respectively. Associated with each bar is a circle whose size encodes the cost of the operation. Note that circle positions do not encode any data attributes. Figure 6 shows house sale data. It also contains interval bars which show the *selling-price/asking-price* range for house sales, and a set of circles whose x-positions indicate the *agency-estimate* attribute. The color of the bars encodes the house neighborhood. Note that the interval bars and the marks in Figure 6, unlike in Figure 5, are not attached to each other positionally. Therefore, it is necessary for search techniques to be sensitive to the subtle difference in the relation between the circle and bars in the two figures.

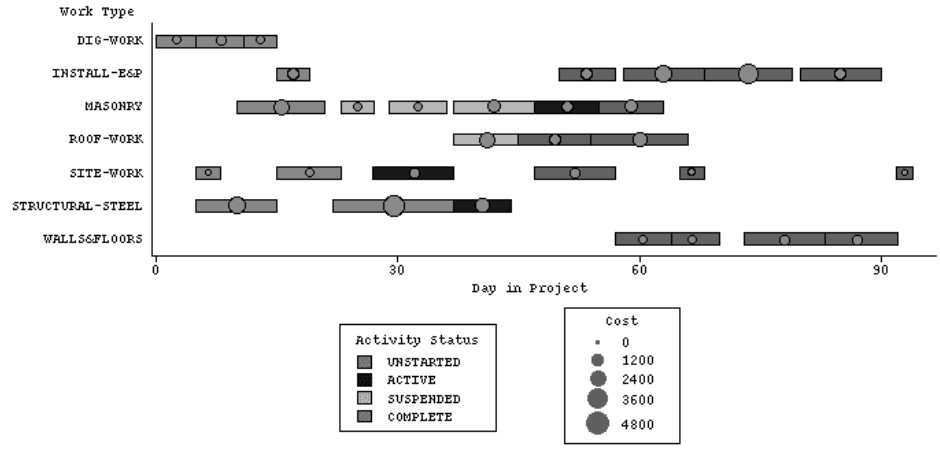

Figure 5: A data-graphic using interval bars and a mark associated with each bar

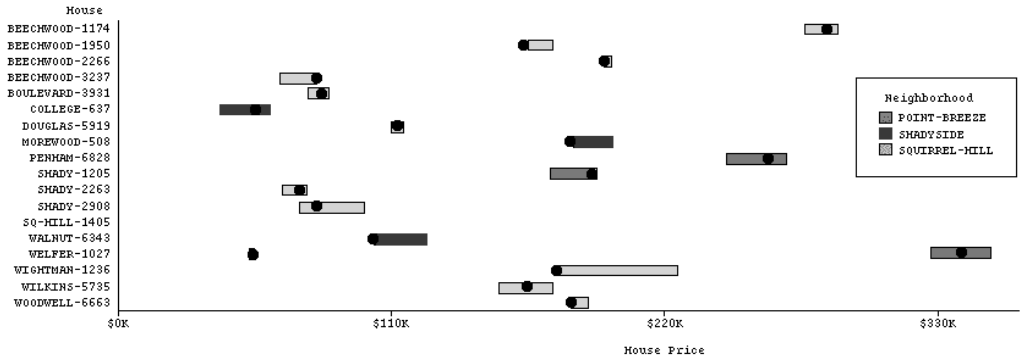

Figure 6: A data-graphic using interval bars and marks whose x-positions encode a data attribute

4.2 SageBook Exact and Similarity Search

SageBook provides matching based on both graphical elements and data attributes. There are several alternative match strategies which provide different degrees of relaxation on the search criteria based on the degree of overlap between the library data-graphic and the user query. Each retrieves a different number of data-graphics depending on its degree of relaxation.

A typical reason for relaxation is to find compromises if no entries match the query exactly. Additionally, similarity-based relaxation finds items that are equally desirable but would otherwise not match because of insignificant feature differences. Most importantly, supporting data-graphic design suggests an additional function of relaxation: giving users ideas for how to integrate additional graphical elements and properties with partial designs they have created. The latter answers questions like: "How can additional *graphemes* be added to the space I've created and integrated with the graphemes I've already included?"; "How have previous data-graphics used additional properties of these objects?"; and "How can other *spaces* or *graphemes* be substituted for the ones I've selected to express the same data?" Enabling users to answer questions like these motivated the choice of match criteria that evolved in SageBook. Finally, our choice of criteria reflected the fact that it was easy for users (or the system) to remove extra *spaces, graphemes* and *properties* when the design is adapted for new data. In the future, we intend to extend the model by taking user tasks into account and providing matches based on task specifications.

4.2.1 Graphical Matching

Close Graphical Matching. This strategy searches for pictures that have the same number of spaces as the query. Figure 7 shows a graphical query and the data-graphics returned for that query. For rhetorical purposes, we have drawn a black border around the seven data graphics that are retrieved by the *close matching* process. All these data-graphics only contain one space because the query only has one space.

For a space in the query to match a space in a library data-graphic, the layout disciplines of both spaces must be identical. However, the library data-graphic space may have more graphemes in it as long as the ones specified in the query match graphemes in the library data-graphic space. For two graphemes to match, they must be of the same class (i.e. bars, lines, marks) as well as use the same properties (i.e. color, shape, size, width) to encode data. In Figure 7, all the data-graphics returned by this search strategy have a space of type chart, consistent with the query. Furthermore, all the charts in the resulting data-graphics contain graphemes of type *horizontal interval bar*. Note also that only the positional properties of the bar were specified in the query. If additional properties were specified, these must also be

present in the retrieved data-graphic. The retrieved data-graphics may contain grapheme properties not contained in the query.

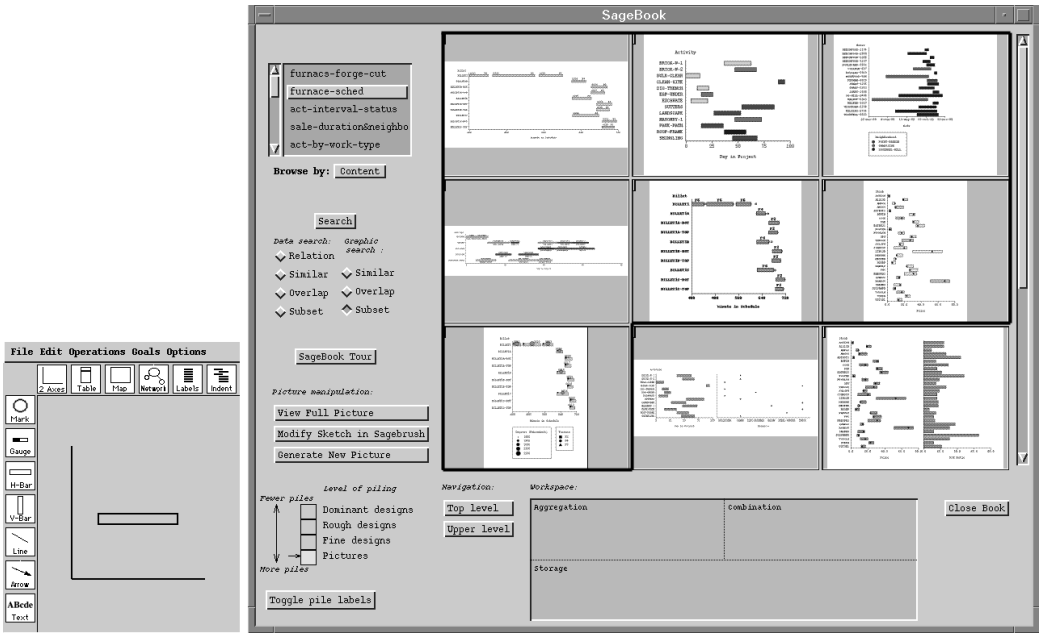

Figure 7: A graphical query and the data-graphics returned by SageBook for that query

Subset Graphical Matching. Subset graphical matching is more inclusive than close graphical matching. In subset matching the data-graphics may have more spaces and graphemes than the query as long as all the spaces in the query match some unique space in the library data-graphic. The data-graphics shown in the grid in Figure 7 are returned by subset graphical matching. The matches returned are sorted according to their degree of similarity to the query, based on the match criteria. For example, in Figure 7 all one-space matches are shown first, followed by the two-space matches and so on.

The subset matching idea is similar to the process of a library search. First, the user enters a query, and a super-set of data-graphics is returned. Each data-graphic in the set contains all the elements specified in the query and may contain more. To narrow down the choices presented by subset matching, the user may have to add some additional constraints to the query. Alternatively, the user may browse through the data-graphics and pick one based on other criteria. Even though the data-graphic retrieved by subset matching may contain more

spaces than the user desires, the unwanted spaces can easily be edited out of the data-graphic by using SageBrush.

Overlap Graphical matching. Subset matching may sometimes exclude data-graphics that are useful but fall slightly short of meeting the match criteria (which requires each space of the query to match with at least one space in the library data-graphic). Thus, in addition to a strict subset search, we implemented a match strategy that sets upper and lower bounds around the number of spaces in the query that must match a space in the library data-graphic. The lower and upper bounds are set to be a percentage decrement and increment of the total number of spaces in the query.

4.2.2 Data Matching

Data-Relation matching. Relation matching retrieves all previously saved data-graphics that have all of the desired relations specified in the query. By relation we are referring to the name of the relation scheme in relational database terms. This matching strategy is useful for retrieving sets of daily or weekly data from the same database and redisplaying them consistently. This also suggests an additional use for data-graphic retrieval - searching for information (rather than just graphic displays) stored in graphic media. For example, the *house-sales* relation might be used to construct many displays over time.

Close Data Matching. This strategy enables users to find graphics that contain data that have similar characteristics to their current working set. Given a list of attributes and their properties, the close data matching algorithm tries to find a mapping from attributes in the query to attributes in prior data-graphics. For an attribute in the query to match an attribute in a library data-graphic, the two attributes must have the same data-type (nominal, ordinal, quantitative) and frame of reference (quantitative/valuation, coordinate), as well as participate in the same kinds of functional-dependencies and complex types. Figure 8 shows an example of this data-matching process. In Figure 8, *Activity* matched *HouseID* because both are nominal data-types and both are functionally independent. *Materials-cost* matched *Number-of-Rooms* because both are quantitative data-types and are the same frame of reference (quantitative/valuation). *Start-Date* and *End-Date* match *Date-on-Market* and *Date-Sold* because they are the same frame-of-reference (Coordinate) and belong to the same

complex-type (Interval Type). The close data match process must also ensure that the number of attributes in the query and the library data-graphic are the same and that every attribute in the query matches some unique attribute in the library data-graphic. In Figure 8, the query and library data-graphic satisfy both of these criteria.

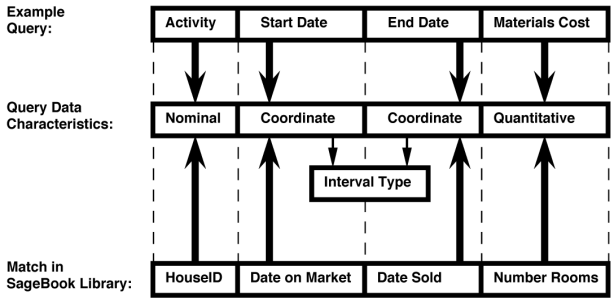

Figure 8: Close data matching process

Unlike the relation matching strategy which requires the query and the library data-graphic to contain identical relation names, the close data matching strategy only requires that the attributes have similar data characteristics. Thus, this search strategy is not a keyword search but a search based on the similarity of characteristics and relationships of the data.

Subset Data Matching. The idea behind subset data matching is very similar to that of subset graphical matching. Subset matching is like close data matching, except that instead of requiring a bijective (i.e. one-to-one and onto) mapping of attributes between the query and the saved data-graphics, subset matching allows the saved data-graphics to have more attributes than the query as long as each attribute in the query has similar characteristics with some attribute in the data-graphic.

Data-overlap matching: As with graphical matching, a variant of the data-subset matching strategy was created that sets upper and lower bounds around the number of data-attributes that must match instead of a strict subset rule.

4.2.3 Manual Search

Users may also manually search through the library. This is necessary in the cases where the user is uncertain how to form a query or unsure what to look for. Manual search is also useful when the user is unfamiliar with the contents of the library and wants to quickly get an idea of what data-graphics are available.

SageBook provides two different library browsing facilities that support manual search. First, users may organize and browse data-graphics through a folder system. This facility allows users to easily navigate through directories and view data-graphic files in the same grid used for visualizing the results of a search (Fig. 7). In addition to supporting manual search, this file navigation interface is also useful when a user knows the exact location or filename of the data-graphic of interest. The file navigation interface provided is similar to those seen on Macintosh interfaces.

The second facility provided to support manual search is a book tour. The tour facility in SageBook includes all of the library data-graphics in a book-like structure. The book consists of chapters, sections, and subsections, which serve to provide a hierarchical categorization to the data-graphics. More detail is provided in the next section on browsing.

5. BROWSING

If the library contains many objects, some queries may retrieve a large set of items. In such cases, the cognitive load placed on the user to browse through the retrieved objects would be significant. In order to deal with large retrieval sets, it is necessary for the system to provide tools to help users browse search results quickly and effectively. Browsing is also useful in those cases where the user simply wants to navigate through the library. Such cases arise when the user is unsure of the kinds of graphics contained in a library, when the user is unable to formulate a query, or when the user simply wants to go through many alternatives to get ideas. Despite its importance, browsing is rarely supported in retrieval systems. Many systems provide a sorted list of objects to the user, making it difficult for users to identify categories or to determine general features and distributions of the result set.

Some browsing methods include sorted lists, visualization of results (Pu and Lalanne, this volume) and structuring the results, for example, through the use of hierarchies and piles (Chuah et al. 1995; Ballay 1994). The sorted list has no structure apart from being ordered according to match scores. This technique is useful when the user is only interested in individual results and not on overall properties of the result set. One big disadvantage is that when match scores are uniformly high or low for a large set of graphics, the sorted list does not provide any useful structure within that set.

Another browsing method involves 2D or 3D visualization of results. This method is very useful for seeing overall distributions of the results and detecting groups that are significantly different. A problem with this method however is that it is not always clear which attributes of the retrieved objects should be viewed or how to compute a numeric score based on similarity. Finally, browsing can be facilitated by organizing the search results in hierarchical or other structures. Pu and Lalanne (this volume) indicated that categorization is a useful method for supporting browsing. As in the previous case, however, this technique suffers from the uncertainty of what are good attributes to use as the organizing structures.

5.1 SageBook Browsing

To support browsing, we developed a scrollable, grid-like interface that enables multiple data-graphics to be viewed at once (Figure 7). Our recent work has explored ways to enhance browsing efficiency by grouping similar data-graphics into a stack in one cell of the grid. Thus the retrieval in Fig. 7 can be compressed to three stacks as shown in Fig. 9. The number of data-graphics in a stack is indicated by the length of a black bar at the top of each cell. For example, the first stack in Fig. 9 clearly contains more graphics than the others. The expand operation can be used to distribute members of any stack into a new grid. An interesting challenge has been to develop effective grouping strategies (i.e. similarity criteria) for organizing a large number of data-graphics into a small number of meaningful stacks. The formal representation of data-graphics provides a framework for regrouping strategies, as it did for graphic and data queries.

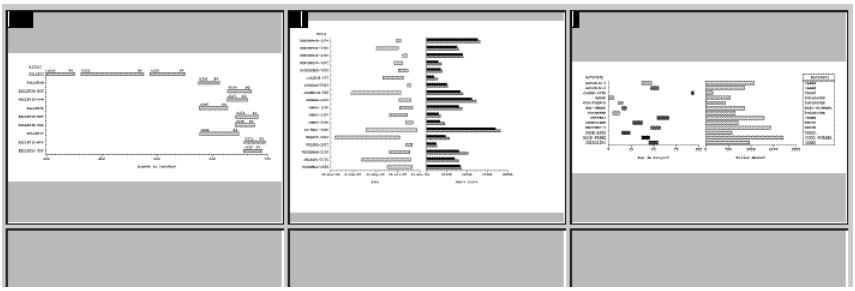

Figure 9: A grid of stacks

We are experimenting with three levels of granularity for grouping graphics: *fine design, rough design*, and *dominant design*. Since SageBook's purpose is primarily to help users get design ideas, these levels are intended to increase the design differences between stacks by grouping similar data-graphics together. *Fine design* includes together those data-graphics that have the same number and types of spaces, number and types of graphemes within each space and properties of graphemes. Effectively, these are cases in which the same design was saved for different data. Viewing the same design with multiple data sets can often bring out striking differences in how effective it is for different contexts.

The *rough design* method differentiates data-graphics based on the space types and grapheme types, and not on the properties of those graphemes. This method I similar to the fine-design criteria except that graphemes may use different properties and the number of each grapheme type in a space may differ. For example, data-graphics like the ones in Figures 5 and 6 would be stored in the same stack regardless of the properties of the graphemes that were used (e.g. circle size). In addition, this also groups bar charts with one, two or more bars per axis element in the same stack or maps with points containing a single label or multiple labels in the same stack. The intent of this type of grouping is to group together graphics that share basic design elements that differ mainly in the number and properties of graphemes they contain. For example, bar charts containing a single bar per independent variable would be included with charts containing two or more bars. Charts that contain bars with superimposed circles would be differentiated from charts that have bars with text labels because they emphasize different design relationships.

Finally the *dominant design* method further relaxes the constraints on the data-graphics that go into the same pile. It requires that the data-graphics that are grouped into one pile have the same spaces and each space has the same *dominant* grapheme. The dominant grapheme is the most salient grapheme in a particular space. For example, lines or bars are dominant when they are combined with marks. This technique was developed when it was discovered that users frequently searched for use of particular graphemes and had biases as to which graphemes were primary or secondary.

We have found that these methods significantly lower the number of data-graphic piles that are shown to the user at any one time. For example, the retrieval of 15 graphics for the horizontal interval bar query (9 of them are shown on the grid in Fig. 7) was compressed to 3 dominant grapheme stacks (Fig. 9). The degree of reduction will depend greatly on the nature of the library. Libraries created by individuals who store many examples of the same graphic will show much different savings from libraries specifically constructed to maximize differences among graphics to provide many design alternatives.

Another metaphor for browsing is that of a *book*. Search results are organized into a book organized into chapters, subchapters and sections based on a classification scheme. Users can browse this book at any level, including individual graphics. Part of the browsing hierarchy is shown in Figure 10. At the first level, data-graphics are divided according to the space disciplines they contain, namely charts, maps, tables, etc. In a study conducted by Lohse (Lohse et al. 1994), these were the main categorizations picked by the users. The next level in the hierarchy separates out data-graphics by grapheme class within the same space type. This is the next most salient feature. Data-graphics that have multiple grapheme classes will be present in multiple piles, enabling people to access them from different perspectives. Finally, we classify the data-graphics according to the properties used. Note that, as before, if a data-graphic is using two different properties, it will be classified under two different sections in the hierarchy.

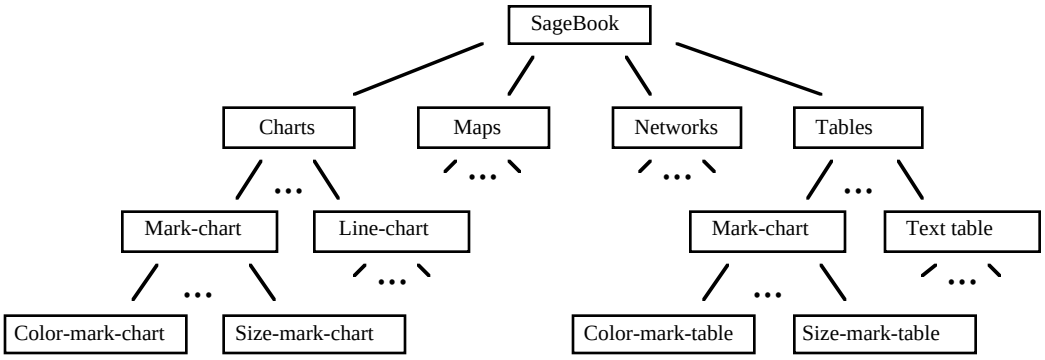

Figure 10: The book organization

The browsing interface also provides a workspace where collections of data-graphics can be stored and manipulated by users. This can be achieved by dragging individual or piles of data-graphics into the workspace. Within the workspace, users may organize their retrievals into new piles by merging, differentiating, and intersecting sets of graphics designs.

6. ADAPTATION

The most important assumption of SageBook is that graphics are retrieved to support design. Therefore, while the quality of the retrieved set is important, what is equally important is how easily an element of this set can be modified to create the desired design. Relatedly, the use of *similarity* search strategies opens up the possibility that some of the graphics retrieved may not fully conform to what the user desires. In addition, users may sometimes need to edit the results to combine multiple retrieved objects. To support these tasks, retrieval systems that support design need to provide tools that facilitate adaptation of retrieved objects. Most systems do not support adaptation, and the few that support it usually only provide manual editing capabilities. The adaptation process, however, may sometimes be complex and inexperienced users or designers may have trouble articulating desired effects. SageBook provides users with automatic adaptation as well as manual editing capabilities. The automatic adaptation facility alters prior designs to suit current data and design preferences.

6.1 SageBook Adaptation Support

To support adaptation, our system provides manual adaptation capabilities with SageBrush and automatic adaptation capabilities through SageBook. The automatic adaptation module maps data-attributes in the query to data-attributes in the retrieved data-graphic based on their similar data characteristics. This mapping process can produce two types of inconsistency: (1) The retrieved data-graphic has more graphical elements than necessary to express the data-attributes in the query. This can result from a *subset* query that retrieved a graphic portraying more data attributes than were in the query. (2) The graphical elements in the retrieved graphic do not express some of the data-attributes in the query. This can happen when an *overlap query* retrieves a graphic that does not satisfy all the query attributes. It is possible for both inconsistencies to occur simultaneously.

In the first case, the adaptation module will discard extra graphical elements in the retrieved data-graphic. The module first discards any extra properties of graphemes (e.g. discarding the use of color or size). If discarding these properties results in a grapheme that would have no properties mapped to data, then the grapheme itself is discarded. Finally, if a space contains no remaining graphemes, it is discarded as well. The progressive removal of unnecessary graphical elements preserves the most salient features for possible use. During adaptation, we assume that the user would want to change the retrieved design as little as possible when there are still data attributes to be expressed. Because a space is a very salient design feature, it will be very noticeable to the user when the adaptation module deletes a space from the design. Similarly, removing graphemes is more salient than not using some of their properties. In the second case, when there are data attributes that are not expressible by the graphic, SAGE will search for an additional design to express them.

Figure 11 shows a data query for attributes of house sales and an example retrieved graphic that shows attributes of construction project activities. The graphic was retrieved because the *start-* and *end-date* interval in the graphic matches the *asking-price/selling-price* range in the query, the *activity-name* attribute matches the *house* attribute. However, the *activity status* attribute, expressed by the color of the bar, does not match the *agency estimate* attribute (the

latter is a quantity: dollar amount, while the former is a set of four nominal values). Since the search criterion was overlap, the graphic was retrieved even though these two do not match.

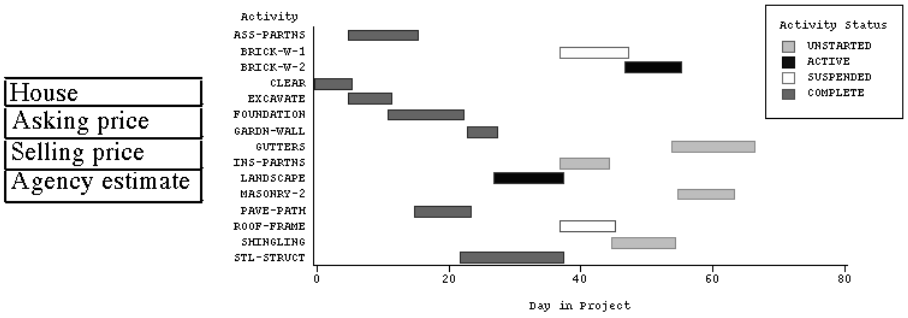

Figure: 11. A data query and an example data-graphic returned by that query

To adapt Figure 11, the color property is first discarded because it does not match any data. Then, the adaptation module sends the query data and partially completed graphic to SAGE to search for an additional design element to complete it. Figure 12 shows the new data-graphic.

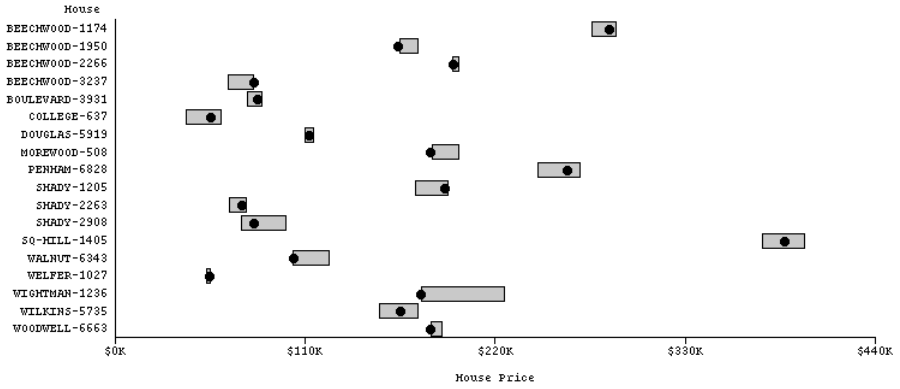

Figure 12: A new data graphic generated from the query and the data-graphic design in Fig. 11 after automatic adaptation

SAGE encoded *agency-estimate* by using the x-position of a new mark grapheme. Position is an effective property for representing attributes and is preferred over size, shape, color, text

lettering, etc. However, the positional properties of the bars are already used, so SAGE added a new grapheme to the chart space. In this case, the additional attribute was a dollar-amount and could be expressed relative to the same x-axis as the other dollar-amounts. Otherwise, SAGE would have to choose between adding another chart aligned to the right of this one or using text labels on the bars to express the agency estimate. In general, SAGE can generate multiple alternatives from which a user can choose. We have encoded a knowledge base of design techniques within SAGE that can complete partial design specifications or create graphics when no specifications are provided (Roth et al. 1994).

7. EVALUATION

Common evaluation measures such as precision and recall have been used to indicate the performance of a retrieval system. Generally, performance is affected by three classes of errors depending on which part of the system they are produced (cf. Fig. 1): interface errors (that affect query construction and adaptation), translation errors (i.e. from query to internal representation), and retrieval errors (that occur in search and match steps).

Interface errors are user-system communication problems that might occur when users are unable to accurately convey their queries or adapt retrieved objects for their current goals using the SageBrush interface. In principle, these errors may occur as the result of poor editing tools. As a result of iterative interface improvements based on preliminary user tests interface errors have been reduced to insignificant levels. The main remaining weakness is the potential for users to specify complex spatial relationships among graphemes that are not yet accurately interpreted by the system (e.g. the arrangement of several text labels around a bar in a chart). Problems like these indicate incompleteness in the representation language and greater flexibility in the interface than is supported by the representation. It can be solved by extending the language or constraining the interface. Related problems can occur in manual adaptation.

Translation errors occur during the conversion of raw objects and queries into the system's internal representation. They usually occur when the internal structure is very high level because complex embedded semantic information has to be derived. For example, translation errors are especially problematic for image retrieval systems where high-level semantic

information embedded in the image needs to be extracted accurately (Griffioen, Yavatkar and Adams, this volume).

SageBook does not have translation errors because the query interface is tightly coupled with the internal representation. SageBook has access to the model descriptions of the data-graphics as generated by SAGE during graphics creation. These storage descriptions contain at least as much information as the data-graphic itself; therefore, there are no inconsistencies between the generated data-graphic and their stored descriptions.

Retrieval errors fall into two categories: structural errors and relevance errors. Structural errors occur when the match criteria does not coincide well with user expectations. Since the user's model is usually at a high semantic level, these errors tend to occur when the internal representation is at a very low level. For example, if the representation contained image properties such as object outlines, then objects that users perceive as structurally identical might not be correctly matched due to orientation and scale differences. Pentland, Picard and Sclaroff (1994) described such matching errors. SageBook has few structural errors because of closeness of the query language to the system representation.

The other type of retrieval errors are *relevance errors*. The concept of relevance reflects the degree of utility of the retrieved objects for users and the tasks they perform. For example, if a user can easily adapt a visualization from the library to his or her data, then we can assume that that visualization has been relevant to that user. However, since designs are so task oriented, just creating a new visualization from an old design may not be sufficient for judging design relevance: the new visualization might be appropriate for some tasks and not others (e.g. viewing correlations vs. making comparisons between pairs of values). Conversely, a library design may not be directly adaptable to a new visualization but it may contain just one element that is valuable for the user. By using this element, one might be able to solve a difficult design problem. In the latter case, even though the user would discard a major part of the original design, it still should be considered relevant. These characteristics of designs make the evaluation of our retrieval and automatic adaptation techniques difficult and perhaps suggests that traditional information retrieval evaluation methods may be

insufficient. New research must create an evaluation procedure for judging the relevance of designs that is based on:

- a method of assigning a degree of relevance based on measures of the ease and completeness of the supported design task (including feedback from users on the value of the retrieval set for the tasks);
- well populated libraries collected from naturally occurring samples;
- a corpus of naturally occurring data analysis tasks to be performed with the search task.

8. CONCLUSION & FUTURE WORK

We have presented querying, storing, searching, browsing, and adaptation techniques pertaining to the retrieval and reuse of graphic designs. We also describe an implemented system, SageBook, that employs these techniques. The following types of retrieval components are essential for this domain:

- a common vocabulary for representing data-graphics designs;
- an object manipulation querying interface (like SageBrush);
- graphic- and data-matching strategies that support different levels of similarity between queries and designs;
- a visual interface for presenting the results of the retrieval and flexible browsing at different levels of aggregation;
- automatic and manual adaptation of retrieved designs for new data.

Preliminary experience with SageBook suggests that these features ensure very few interface, translation and structural errors. Positive evaluation of relevance errors depends both on methods for assessing tasks, as well as encoding task properties in the query and search process (i.e. enabling users to express the tasks they wish to perform with retrieved graphics). Future work includes performing comprehensive user testing and incorporating SageBook as a design support tool within an extensive information exploration environment called Visage (Roth et al. 1996).

9. ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of Mark Derthick, John Kolojejchick and Joseph Mattis to the research, implementation and preparation of this paper. This work was supported by DARPA and Army Research Lab.