A Model for the Visualization Exploration Process

T.J. Jankun-Kelly* Kwan-Liu Ma* Michael Gertz†
Computer Science Department
University of California, Davis

ABSTRACT

The current state of the art in visualization research places a strong emphasis on different techniques to derive insight from disparate types of data. However, little work has investigated the visualization process itself. The information content of the visualization process—the results, history, and relationships between those results—is addressed by this work. A characterization of the visualization process is discussed, leading to a general model of the visualization exploration process. The model, based upon a new parameter derivation calculus, can be used for automated reporting, analysis, or visualized directly. An XML-based language for expressing visualization sessions using the model is also described. These sessions can then be shared and reused by collaborators. The model, along with the XML representation, provides an effective means to utilize the information within the visualization process to further data exploration.

CR Categories: I.3 [Computer Graphics]: Methodology and Techniques; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Theory and Methods

Keywords: visualization process, visualization models, visualization systems, scientific and information visualization, collaboration, XML

1 INTRODUCTION

Over the past decade, the field of visualization has matured; a wealth of techniques for a variety of data types have been developed to solve problems in various domains. As the use of visualization becomes more wide-spread, a formal understanding of how visualizations and visualization systems are used is needed. Such a formal understanding can assist in the development of new visualization systems or the refinement of current ones. For example, a system using a complete model of the visualization process can suggest courses of data exploration for users by analyzing previous results captured by the model during the current or previous exploration sessions. In addition, a standard representation of the visualization process provides documentation of that process. This documentation can be used by others to reproduce the visualization results for validation or to extend those results by continuing the data exploration. Towards this end, a model for the visualization exploration process has been developed.

*Visualization and Graphics Research Group, Center for Image Processing and Integrated Computing, Computer Science Department, University of California, Davis, CA 95616. E-mail: `{kelly, ma}@cs.ucdavis.edu`

†Database and Information Systems Group, Computer Science Department, University of California, Davis, CA 95616. E-mail: `gertz@cs.ucdavis.edu`

It is important to note that a model of the visualization process alone is not sufficient to describe the knowledge of the user before or after the visualization. To completely capture this knowledge and insight, a meta-data model for the visualization process also needs to be developed. The ultimate goal of this research is to develop such a meta-data model using the model described here as the basis for the meta-data's descriptions. For example, meta-data annotations of previous sessions that suggest what results were "good" and which were not could help the session analysis process. Thus, this work is the first step towards that goal.

There are several benefits to capturing the visualization process with the proposed approach. With our process model, users of visualization systems are able to record their visualization sessions at a higher level than simple log systems are able to provide. For example, the two representations of the visualization process discussed in Section 3.2.1 would be difficult or impossible to create using a log file due to the lack of information about the process. Our model also allows visualization systems designers to build systems that can share results and process information between different visualization interfaces, an example of which is discussed in Section 5. Finally, represented with a formal model, the visualization process is opened up to a variety of analyses. Though work of this nature is not presented here, it is conceivable that analysts could derive different metrics using the model to further examine and optimize the visualization process. The process model discussed here addresses visualization exploration in more depth and with greater generality than has been previously presented.

2 A CHARACTERIZATION OF VISUALIZATION EXPLORATION

In order to develop a model which describes the visualization process, the characteristics of that process must be understood. Springmeyer et al. [20] describe the entire scientific data analysis process of which scientific visualization is a part. In their taxonomy, visualization is used mainly to interact with and maneuver through scientific data. These actions include generating, examining, querying, navigating through, comparing, and classifying the data or portions of the data. While this task-based classification is useful for understanding the uses of visualization, it is less useful in distinguishing the core features of the visualization process.

Upson et al. [22] describe the scientific visualization process as an iterative analysis cycle. According to this model, data is filtered into subsets of interest, mapped onto visual primitives, and then rendered for the user by a function called the *visualization transform*. The visualization generated by this transform is then used by the user to provide feedback into the previous steps, restarting the cycle. A similar cycle of raw data transformation, visual structure generation, and view rendering with user interaction is described by Card et al. [5] for information visualization. The key feature of both models is that the visualization process is an iterative sequence of user controlled transformations. Thus, elements that change during this iteration must be the focus of any description of the visualization process. These elements are the parameters which control

Type	Parameter Interactions	Continuous Rendering	Transform Editing
Interactive Parameter Control	Single or multiple parameter editing	No	None or limited
Dynamic Manipulation	Single parameter editing	Yes	None or limited
Data-flow	Single or multiple parameter editing	Maybe	Full
Image Graph	Single parameter editing; parameter propagation to linked results; operators to create new parameters	No	None
Exploration Spreadsheet	Create, edit, or remove a single parameter; operators to create new parameters; interpreter to manipulate parameters	No	None

Table 1: A summary of parameter editing capabilities of visualization exploration interfaces. The first column reports the type of interface, the second column the different type of parameter manipulation the interface supports, the third column whether the interface supports continuous rendering from parameter updates, and the fourth column whether the interface supports visualization transform editing. A "limited" in the fourth column signifies that the interface may combine a static number of different visualization techniques, such as a cutting plane in a volume rendering display.

the transforms. As parameters are visualization transformation dependent, a description of this transform is also important to any documentation of the visualization process.

To gain a better understanding of the visualization process, it is insightful to investigate different visualization user interfaces. The kinds of interactions a user has with the user interface expose different aspects of the visualization exploration process. Though the user interface community has extensively looked at user interface events (for example, see [10]), the actions discussed here are at a higher semantic level and are tailored for visualization systems. Table 1 summarizes the five interface types studied and different capabilities of each. The examples discussed below focus on scientific visualization interfaces, though parallels can be drawn to information visualization tools.

The first two interface types discussed are interactive parameter control and dynamic manipulation interfaces [19]. In the former interface, interactive manipulation of the parameter values does not correspond to interactive updates to the rendered result; in the later interface, the result is rendered interactively during parameter changes. Visualization transform editing—the process of creating visualization transforms through means such as data-flow networks (c.f.)—is not supported in these interfaces, though a fixed number of different transforms may be available for use. In these systems, the major action is the editing of parameter values (potentially from different parameter types in the case of interactive parameter control interfaces) to generate a visualization result. In interactive interfaces, parameter values can vary over a continuous range during manipulation. This range corresponds to a range of rendered results, which must be documented somehow. This behavior does not occur in non-interactive interfaces.

Data-flow interfaces [1, 14, 22, 23] provide the ability to edit the visualization transform. The visualization transform is constructed using a visual programming language. Because the parameter types are dictated by the visualization transform, special care must be taken when recording the visualization process for data-flow interfaces. Depending on the system, parameter changes and rendering may be synchronous like dynamic manipulation interfaces or asynchronous like interactive parameter control interfaces.

The fourth type of interface presented here is the Image Graph [17]. This interface uses a graph representation of the visualization process that distinctly displays the relationship between generated images via glyph edges. An image graph interface exhibits two types of parameter manipulation behaviors not found in the previous interfaces. First, it is possible to propagate a parameter change down a directed graph of related results, creating an entirely new subgraph of images. In addition, it is possible to combine several parameters from different results via parameter operators to generate new parameters and thus a new result.

The interfaces discussed so far have coupled parameter editing and rendering. Though it may be possible to edit several parameters without generating a result—and even change the same parameter multiple times before rendering—only parameters which generate a result are stored. This behavior is not true of Jankun-Kelly and Ma's visualization exploration spreadsheet-like interface [11]. In this interface, parameters can be added, edited, or removed without rendering a result. When desired, the user combines a set of parameters—represented by a cell in the table—to generate a result. While all the interfaces perform the same function—collecting parameter values to create the corresponding visualization result—the spreadsheet-like interface highlights this behavior.

From the previous discussion, several key properties of the visualization exploration process can be distilled. As illustrated by the spreadsheet interface, the fundamental operation of the visualization exploration process is the application of a set of parameter values to the visualization transform to generate a visualization result. The parameter values are generated in one of three ways: a single parameter value can be generated from an old parameter value (as in non-interactive interfaces); a range of parameter values can be generated from an old parameter value (as in an interactive interface); or a set of parameter values can be derived from a different set of parameter values via some operator (as in the image graph). The parameters available vary with the visualization transform which can also change. Finally, it is possible to generate more than one result in a single operation in some interfaces. All of these properties will be addressed by the proposed model for the visualization exploration process.

3 VISUALIZATION EXPLORATION PROCESS MODEL COMPONENTS

The visualization exploration process model consists of two components: the visualization transform model and the visualization exploration session model. The transform model provides context for the process described in the exploration session model. Put another way, the visualization transform model describes how the visualization occurs while the visualization exploration session model describes what occurred.

3.1 Visualization Transformation Model

The visualization transform model outlines the type of visualization being applied. Without this information, it may be difficult to determine how the visualization results were generated. The transform also details what parameter types are used to generate a result. This information is needed by the session model in the next section.

Previous visualization transform models have focused on describing the visualization transform itself, leaving out descriptions of the parameters involved. One model already mentioned is the data-flow model used by data-flow interfaces [8]. This considers the visualization transform a pipeline where each stage in the pipeline represents a transformation. When these stages are collected, they form a network through which data flows. The data state model [6], in contrast, focuses on the transformation of data states through the visualization pipeline. In a data state network, the nodes represent states of the data and the edges operations on the data. This network is the dual of the data-flow network. Finally, the display model used by VisAD [9] is interesting in that it maps data attributes directly to display via mappings utilizing lattice theory and principles developed by Bertin [2] and Mackinlay [18]. It is the lowest-level approach of the three models. These models neither explicitly address the parameters encoding the visualization technique nor describe the visualization process as a whole.

The visualization transform model described here augments the data state model to include information about the parameters used in the visualization process. The data state model was chosen because it already focuses upon the data sets utilized in the visualization, an important parameter. Formally, the transform model consists of the following components:

Visualization transform A function $f : D \times P_1 \times \cdots \times P_n \rightarrow R$ which describes the mapping of value (the data set type D) to view (the visualization transform result type R). P_1 to P_n are *visualization transform parameter types*.

Visualization transform parameter type Any set that is part of the domain of the visualization transform function. By this definition, data set types are also a visualization transform parameter type. A member of a visualization transform parameter set is a *visualization transform parameter value*, or parameter value for short.

Visualization transform result type A set which is in the range of a visualization transform function. Members of this set, known as *visualization transform result values*, or results for short, are directly representable in graphical form (such as a raster image, shaded geometry, etc.).

When documenting the visualization exploration session, only the visualization transforms and corresponding parameter and result types used need to be recorded.

3.2 Visualization Session Model

The visualization exploration session model serves two purposes. Its primary purpose is to capture the path of exploration during the visualization session. This information encapsulates the details of the visualization exploration process. The secondary purpose of the model is to allow the description of the session to be further analyzed, visualized, or manipulated.

The visualization exploration session model describes four components:

- The visualization results generated during the visualization process.
- The parameter values used during the process.
- A linear history of the generated results.
- An encoding of the relationships between results.

The visualization results are recorded because they are the desired outcome of the user's exploration process. Each result is uniquely identified by the parameter values which generated that result. Without the parameter values, the final results could not be reproduced. Finally, the history and derivation information is needed to reproduce the visualization process.

Most research in visualization modeling has not focused on the process model. One work of note is that of Lee and Grinstein [16], later expanded in Lee's thesis [15]. Lee uses a graph-like structure to model the visualization process for databases. Vertices in the graph represent the state of the visualization while edges are relationships between states. These relationships are based upon similarities between meta-data attributes of the states. In Lee's work, these attributes describe structural attributes of the states/results. Our work also uses a graph structure to represent relationships, but the relationships are between parameter value derivations.

Similar work in process modeling was addressed by the GRASPARC project [4]. The project's work addresses modeling the search for a solution in a scientific problem solving environment (PSE). Like the visualization exploration process, this search is parameter driven; in this case, the parameters are the control variables for the simulation data in the PSE. A history tree structure is used to communicate and manipulate the PSE control state where nodes in the tree store the solution parameters and complete or partial results (as simulations can be interrupted). Our work differs from the GRASPARC work in several ways. First, as noted, their work focuses on the steering aspect of a problem solving process, while we address the search of the visualization parameter space. Second, our work models how the parameter values can change during the visualization process. This information is not present in the GRASPARC work. In addition, parameter derivations in our model can have multiple sources/parents, whereas branches in the GRASPARC history tree are limited to single parent (this is illustrated later in Figure 3). Finally, the derivation information in our model allows the visualization process to be analyzed in many different ways instead of the single, tree-like view GRASPARC provides.

As previously stated, the fundamental operation that occurs during the visualization process is the formation of parameter value sets to derive visualization results. These parameter value sets, or *p-sets*, posses a parameter value for each parameter in the visualization transform. New p-sets are created by user interaction with the visualization system. The session model tracks the relationships between results by recording how a user generates new p-sets (and thus results) from old p-sets. P-set derivations can be expressed as one of three templates using a *parameter derivation calculus*. In the following, the $p_j = \{p_j(1), \ldots, p_j(n)\}$ are p-sets and each $p_j(i) \in P_i$ is a different parameter value for the same parameter type P_i:

I. $p_2(i)|\, p_0 \mapsto p_1$: parameter value $p_0(i) \in p_0$ is replaced by $p_2(i) \in p_2$ in order to derive p-set p_1.

II. $[p_0(i), p_1(i)]|\, p_0 \mapsto p_1$: a continuous range of parameter values is generated between discrete parameter values $p_0(i)$ and $p_1(i)$ and applied to p-set p_0. p_1 represents the p-set at the end of the continuous interaction.

III. $p_0(i) \rightarrow p_1(i)|\, p_0 \mapsto p_1$: parameter value $p_1(i)$ was calculated from $p_0(i)$ by some function and then applied to p_0 to generate p_1.

All derivations are expressed as *parameter-list* | *input-tuple* $\mapsto$ *output-tuple*. Such constructions are *parameter derivation calculus instances*. The parameter list contains input and output parameter values; the former is used to derive the later. Output parameter values are sequentially applied to the elements in the input p-set tuple to generate the output p-set tuple. In the templates, $p_0(i)$ is an input parameter value while $p_1(i)$ and $p_2(i)$ are output parameter values.

It is possible to have multiple input parameter values, output parameter values, p-sets in an input tuple, or p-sets in an output tuple. For example, Figure 1 demonstrates a function derivation with two output parameter values (in braces) and two elements in the output tuple (in angle brackets).

The parameter derivation calculus describes all the salient parameter behaviors described in Section 2. The only operation not supported is the "remove parameter" function of the exploration spreadsheet. This model records how results were generated during the visualization process—thus, only parameters which are used must be recorded. Parameters that were added and then removed without rendering a result have no bearing on any generated results and parameters later hidden from display by removal will still be present in the model's representation.

The parameter derivation calculus is the basis for recording the visualization exploration session. Formally, a visualization session consists of a set of *visualization session results*. A visualization session result is a tuple containing a p-set, the visualization result derived from the p-set, a timestamp to place the result in temporal context, and a parameter derivation calculus instance detailing how the result was derived. Example session results are given in Figure 1. Each session result represents the generation of a single visualization result. As the example illustrates, it is possible for parameter calculus instances to be the same for two or more session results (the third and fourth lines in the example). This disparity is due to the fact that calculus instances correspond to user actions while session results correspond to rendered results. The same user action can create more than one rendered result, all sharing the same timestamp. Though it is possible for a user to re-visit the same visualization result by generating the same p-set more than once, each is a unique session result identified by a distinct timestamp.

One issue remaining is the change of visualization transforms during a visualization exploration session. Changes of visualization transform are not explicitly encoded in the model. Currently, it is assumed that visualization transforms can be uniquely identified by their signature: the parameter types and result type used in the visualization transform. Thus, a p-set not only uniquely identifies a visualization result but also identifies the visualization transform that generated the result since the parameter types are defined by the transform. Explicit identification of a transform change is not needed since it is implicit in the visualization session result sequence.

$$
\begin{aligned}
s_0 &= \langle p_0, r_0, t_0, \emptyset\rangle \\
s_1 &= \langle p_1, r_1, t_1,\ p_1(2) \mid p_0 \mapsto p_1\rangle \\
s_2 &= \langle p_2, r_2, t_2,\ p_1(1) \rightarrow \{p_2(1), p_3(1)\} \mid p_1 \mapsto \langle p_2, p_3\rangle\rangle \\
s_3 &= \langle p_3, r_3, t_2,\ p_1(1) \rightarrow \{p_2(1), p_3(1)\} \mid p_1 \mapsto \langle p_2, p_3\rangle\rangle \\
s_4 &= \langle p_4, r_4, t_3,\ p_4(2) \mid p_0 \mapsto p_4\rangle
\end{aligned}
$$

Figure 1: A series of visualization session results. A session result is a tuple of a p-set (the p_i), the visualization result corresponding to that p-set (r_i), a timestamp (t_i), and information detailing how the result was derived. In this example, the second session result was derived from the first in the second timestep before the third and fourth results were both derived in the third timestep. Afterwards, the fifth result was derived from the first.

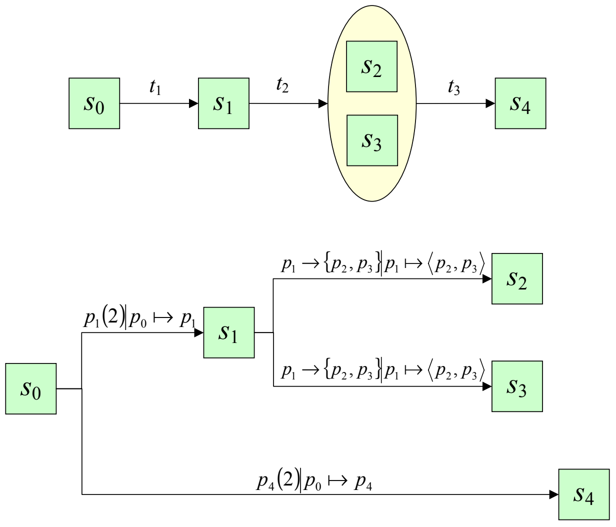

s_0
t_1
s_1
t_2
s_2
s_3
t_3
s_4
$p_1(2)|p_0 \mapsto p_1$
$p_1 \rightarrow \{p_2, p_3\}|p_1 \mapsto \langle p_2, p_3\rangle$
$p_1 \rightarrow \{p_2, p_3\}|p_1 \mapsto \langle p_2, p_3\rangle$
$p_4(2)|p_0 \mapsto p_4$

Figure 2: The history sequence (top) and derivation graph (bottom) for the visualization sessions results (the s_i) from Figure 1. The graphs provide an "at-a-glance" overview of the visualization process. The graphs clearly show that s_4 is not descended from s_2 or s_3, whereas that information may not be immediately apparent from the session results.

3.2.1 Visualization Process Graphs

Visualization process graphs visually summarize the visualization process. Two graphs of interest are the history sequence and the derivation graph (Figure 2).

In the history sequence, branches in the visualization process are collapsed into a single element. Each element in the sequence is a set of session results that were generated by the user in a single operation. The sequence can be displayed graphically using vertices representing the session results created during a single time step and directed edges representing the flow of time. These edges are labeled with the timestamp to order the sequence.

The history sequence is insufficient for describing the relationships between session results. The sequence does not distinguish between results derived directly from their predecessor or those derived from earlier results. These relationships are vital to understanding the entirety of the visualization process and are captured by the derivation graph.

The derived-from relation forms the basis of the derivation graph: A session result s_j is derived from result s_i if and only if s_j's timestamp is greater than s_i's timestamp and the p-set p_i belonging to s_i either possesses a parameter value which is a member of the parameter list in s_j's parameter calculus instance or p_i is a member of the input p-set tuple in s_j's parameter calculus instance. In other words, s_j is derived from s_i if it was created after s_i and either used the p-set or a parameter value from s_i's p-set to derive s_j's p-set. Using the relation, the derivation graph is constructed as follows. Each vertex in the graph represents a single session result. There is a directed edge between two vertices if and only if the vertex with the outgoing edge derived the vertex with the ingoing edge; this edge can be optionally labeled with the parameter calculus information as demonstrated in Figure 2. Derivations that generated or used several results are clearly identified in the graph. It is possible for the graph to contain disconnected components. Each disconnected component corresponds to a different visualization transform as there will be no derivations between transforms. Together with the history sequence, the derivation graph captures the key features of the visualization process.

One property to note of the derivation graph is that it is a collection of directed, acyclic graphs (DAGs). There are no cycles because of the ordering enforced by the derivation relation—no result can derive a result with a lower or same timestamp value. The DAGs represent the derivations related to a single visualization transform. Each DAG possesses a node corresponding to the default result of the visualization transform. The default result is the session result that corresponds to the initial parameter values for a visualization transform; if there is no appropriate default value for a certain parameter (such as a data set), then an "undefined" value is used. By convention, any completely new set of parameter values is derived from this default set. Only the default results are not derived from any other result.

There are potentially other visualization session graphs that can be extracted from the visualization session results. In addition, metrics can be created to measure different properties of these graphs and the session they encode. Lee's thesis [15] describes several such measurements which could be adapted for analysis of visualizations performed with this model. This is a fertile area of research to be explored.

4 MODEL REPRESENTATION

To transport an instance of the model of the visualization process between different systems requires the use of a common data format. To be effective, the format must be extendible to different visualization applications. It is also desirable that the representation can be used by data-mining or analysis tools. These goals are accomplished by using XML to express the visualization process.

The Extended Mark-Up Language (XML [3]) is the standard data exchange format for the World-Wide Web. Standardized technologies exist to parse and extract the content from XML documents. XML documents can also be transformed into HTML [7]. By expressing the visualization session with XML, the session can be easily shared with collaborators. Specific systems can translate their internal representation into our generic model (via XML) which can then be translated again to a representation usable by some other system.

The XML representation of the model is partitioned into five sections. The first section describes the visualization transforms used by listing their signatures (the parameter and result types) and name. In the next section, a list of the parameter values used is stored, each uniquely identified. Each distinct p-set is then recorded by identifying the parameter values composing the p-set. The p-sets are also given a unique identifier. Next, the visualization transform results are stored. For each result, a reference to the p-set which generated it, an identifier, and the result itself are recorded. Note, for interactive systems which can generate results continuously over a range, only the first and last result in that range are stored. It is assumed that the interim results can be generated by interpolating over the parameter that varied. If two results are not sufficient, then "key frame" results—results where interpolation over the range is sufficient—could also be stored. In the final section, the visualization session's results are themselves stored. Each session result identifies its p-set, visualization result, timestamp, and the derivation information for that session result. When generating the XML session document, there are different approaches to how parameters and results are stored. It is possible to embed a representation of these items directly into the XML representation. For large or binary elements, such as the data set used or the results themselves, this approach may be inadvisable. Instead, each parameter or result element in the XML document can provide an optional link attribute. A link is a URL describing where the actual parameter or result may be obtained. Linking can be used to reference large data sets over the network while accessing image files locally, avoiding costly transfers.

Note that the main purpose of the XML description of the model is for transport, not analysis. Analysis is performed on the information encoded by the XML (the visualization session results from this model), not on the XML itself. Given an XML document representing a visualization session, tools are expected to parse the XML into their own internal structures before operating on the visualization session information.

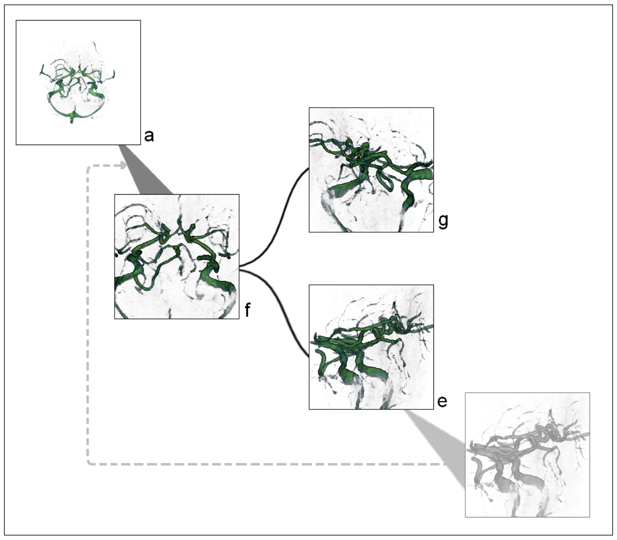
5 EXAMPLES

To better understand the details of the visualization exploration process model, we present a few examples. Two examples are presented in this section. The first example considers a detailed visualization session in order to demonstrate the effectiveness of the model and representation. It also provides a concrete example of how the parameter calculus can describe real visualization sessions. The second example is a case study demonstrating how an implementation of the model was added to an existing visualization tool.

The first example (Figure 3) demonstrates the use of the the model and representation. A volume visualization of blood vessels in the brain was performed using the image graph (top image in the figure). The user first zoomed into a region of interest (result b). Two rotations were then used to display different views of the vessel (c and d). After zooming in again (e), the user decided to apply the final zoom magnification to the earlier images. This was accomplished by dragging the zoom edge over the previous zoom edge. The images using the new magnification (e, f, g) replaced the old images (d, b, and c respectively) to produce the image graph shown in the figure. During the exploration, the session results were recorded (middle portion of the figure); these results explicitly state how the zoom parameter value from e's p-set was applied to results b and c to derive results f and g respectively (the fifth and sixth lines in the middle portion of the figure). The derivation graph (displayed without edge labels in the figure) illustrates this point by displaying how multiple results were used to generate the later results. Finally, the exploration session was stored as an XML document and transformed into HTML (bottom image in the figure). This web page can be easily viewed and shared with collaborators to discuss results.

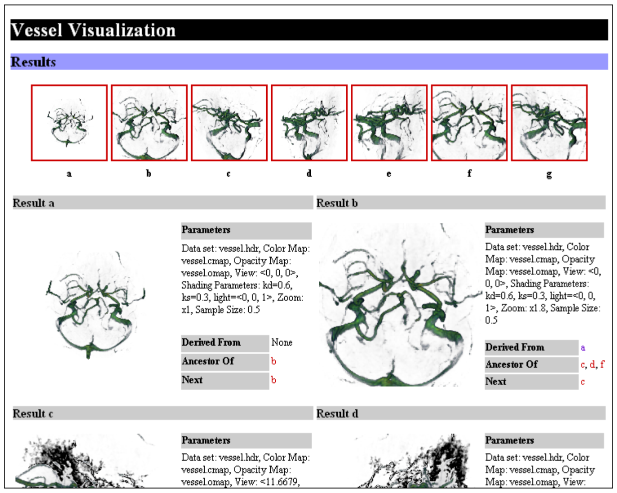
In the second example, the model was used to augment an existing visualization tool. The tool in this example is used to visualize anomalies in Internet routing using the Border Gateway Protocol (BGP) [21]. The tool displays different types of changes to ownership of autonomous systems (ASes)—groups of hosts on the Internet. The different types of changes correspond to different colors (top image in Figure 4). Lines of the appropriate color connect an AS along the edge of the square to IP addresses affected by the AS change within the square. The tool allows a user to browse through different dates with different types of AS changes highlighted. Anomalies are found by visually searching the dates for unusual patterns of lines. In this example session, a range of dates showing all the AS change types were examined until an anomaly was discovered on August 14th, 2000 (top row of the bottom image in Figure 4). The displayed AS change types were then changed until the type of anomaly was isolated (third column in the lower image in the figure).

In order to support the model presented here, the original tool was modified in several stages. First, the types of visualization transforms used by the system and the parameter and result types of each transform were determined. In this system, the major parameter types are the date, which of the eight AS changes to display, a list of ASes to highlight, a list of ASes to ignore, and options for modifying the display. Once the parameters were identified, the next stage determined how the parameter values can change. For this example, all parameter changes are discrete: there are no function applications and parameters cannot be manipulated over a range. Finally, hooks were added to the user interface elements corresponding to each parameter type in order to capture changes in parameter values. These hooks update session information stored by a separate library. The library (written in Python) interfaces with the original tool (written in C++) in order to produce the XML representation used by the spreadsheet (written in Java, the bottom image in Figure 4) to display the process. The above process can be repeated with other visualization systems in order to make use of the model and this XML representation; Figure 5 displays the



$a = \langle p_a, r_a, t_0, \emptyset \rangle$	Default
$b = \langle p_b, r_b, t_1, p_b(zoom) \mid p_a \mapsto p_b \rangle$	Zoom a
$c = \langle p_c, r_c, t_2, p_c(view) \mid p_b \mapsto p_c \rangle$	Rotate b
$d = \langle p_d, r_d, t_3, p_d(view) \mid p_b \mapsto p_d \rangle$	Rotate b
$e = \langle p_e, r_e, t_4, p_e(zoom) \mid p_d \mapsto p_e \rangle$	Zoom d
$f = \langle p_f, r_f, t_5, p_e(zoom) \mid \langle p_b, p_c \rangle \mapsto \langle p_f, p_g \rangle \rangle$	Zoom b
$g = \langle p_g, r_g, t_5, p_e(zoom) \mid \langle p_b, p_c \rangle \mapsto \langle p_f, p_g \rangle \rangle$	Zoom c

$$\begin{tikzcd}
 & & c \arrow[rrd] & & & \\
a \arrow[r] & b \arrow[ru] \arrow[rrr] \arrow[rd] & & & f & g \\
 & & d \arrow[r] & e \arrow[ru] \arrow[rru] & &
\end{tikzcd}$$



Vessel Visualization

Results

a b c d e f g

Result a

Parameters	
Data set: vessel.hdr, Color Map: vessel.cmap, Opacity Map: vessel.omap, View: <0, 0, 0>, Shading Parameters: kd=0.6, ks=0.3, light=<0, 0, 1>, Zoom: x1, Sample Size: 0.5	
Derived From	None
Ancestor Of	b
Next	b

Result b

Parameters	
Data set: vessel.hdr, Color Map: vessel.cmap, Opacity Map: vessel.omap, View: <0, 0, 0>, Shading Parameters: kd=0.6, ks=0.3, light=<0, 0, 1>, Zoom: x1.8, Sample Size: 0.5	
Derived From	a
Ancestor Of	c, d, f
Next	c

Result c

Parameters

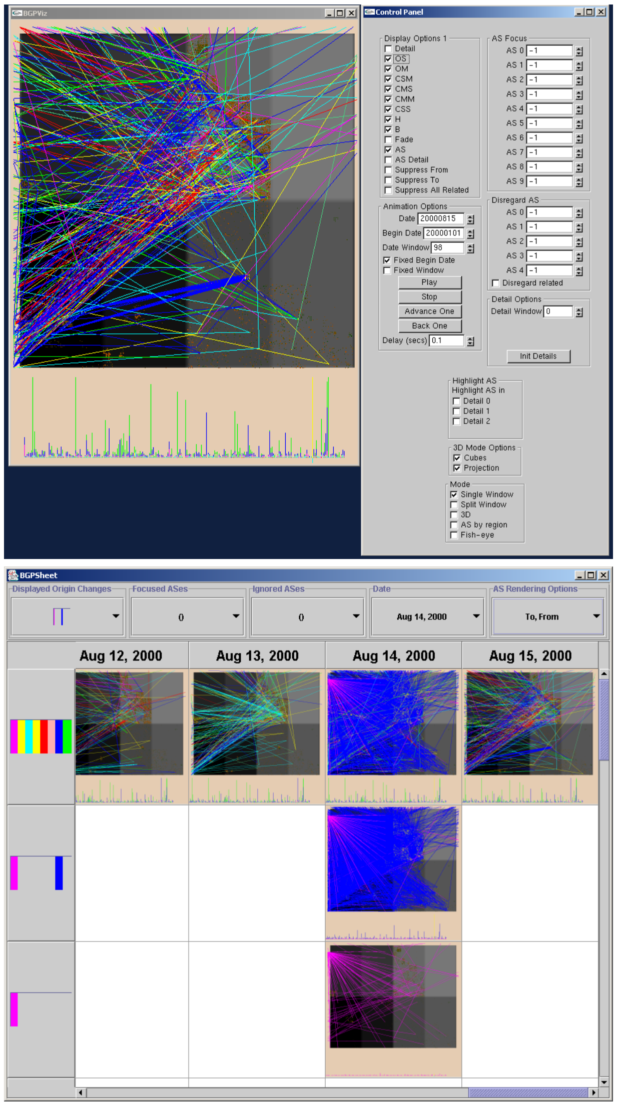
Data set: vessel.hdr, Color Map: vessel.cmap, Opacity Map: vessel.omap, View: <11.6679,

Result d

Parameters

Data set: vessel.hdr, Color Map: vessel.cmap, Opacity Map: vessel.omap, View:

Figure 3: Representation of a brain vessel visualization. The feature of interest is the bulge in the lowest vessel in image e (captions added for clarity). During the visualization, the user dragged the zoom edge going to image e over the edge from a to zoom the other images in the image graph (top). These derivations, including the propagation of the zoom factor from e to results f and g, are recorded in the session results (top middle). The derivation graph succinctly illustrates the information within the session results (bottom middle). The XML representation of the original visualization can be used to translate the session from an image graph view to an HTML page (bottom).



Display Options 1: [ ] Detail, [✓ OS], [✓ OM], [✓ CSM], [✓ CMS], [✓ CMM], [✓ CSS], [✓ H], [✓ B], [ ] Fade, [✓ AS], [ ] AS Detail, [ ] Suppress From, [ ] Suppress To, [ ] Suppress All Related

AS Focus: AS 0 -1, AS 1 -1, AS 2 -1, AS 3 -1, AS 4 -1, AS 5 -1, AS 6 -1, AS 7 -1, AS 8 -1, AS 9 -1

Animation Options: Date 20000815, Begin Date 20000101, Date Window 98, [✓ Fixed Begin Date], [ ] Fixed Window, Delay (secs) 0.1

Disregard AS: AS 0 -1, AS 1 -1, AS 2 -1, AS 3 -1, AS 4 -1, [ ] Disregard related

Detail Options: Detail Window 0

Highlight AS: Highlight AS in [ ] Detail 0, [ ] Detail 1, [ ] Detail 2

3D Mode Options: [✓ Cubes], [✓ Projection]

Mode: [✓ Single Window], [ ] Split Window, [ ] 3D, [ ] AS by region, [ ] Fish–eye

Displayed Origin Changes	Focused ASes	Ignored ASes	Date	AS Rendering Options
	0	0	Aug 14, 2000	To, From

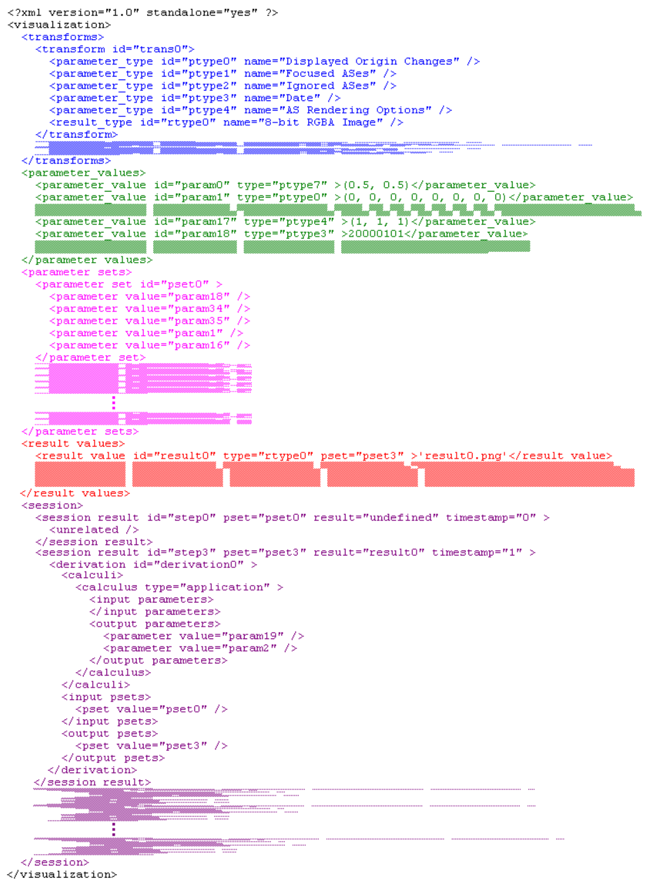
Aug 12, 2000 | Aug 13, 2000 | Aug 14, 2000 | Aug 15, 2000

Figure 4: An example of augmenting an existing visualization system (top) to store visualization session information using the model. The interface is a Border Gate Protocol (BGP) visualization tool; the bottom figure displays a spreadsheet view of an exploration process originally captured by the tool. Dates are shown across the columns and the types of origin AS change displayed (indicated by color) are shown down the rows.

salient portions of the XML. In this example, parameter values are encoded in the XML directly (such as the date January 1, 2000 for "param18"), while results are referenced by local files (such as the PNG image file result0.png for "result0").

6 INITIAL EVALUATION

Initial evaluation of the model and representation have been positive. Users of the BGP tool, for example, were pleased with the ability to view the visualization in different formats by using the system-independent XML representation. Demonstrations of a spreadsheet-based system using the model to display session summaries in HTML have also met with approval. More studies, es-



```
<?xml version="1.0" standalone="yes" ?>
<visualization>
  <transforms>
    <transform id="trans0">
      <parameter_type id="ptype0" name="Displayed Origin Changes" />
      <parameter_type id="ptype1" name="Focused ASes" />
      <parameter_type id="ptype2" name="Ignored ASes" />
      <parameter_type id="ptype3" name="Date" />
      <parameter_type id="ptype4" name="AS Rendering Options" />
      <result_type id="rtype0" name="8-bit RGBA Image" />
    </transform>
  </transforms>
  <parameter_values>
    <parameter_value id="param0" type="ptype7" >(0.5, 0.5)</parameter_value>
    <parameter_value id="param1" type="ptype0" >(0, 0, 0, 0, 0, 0, 0, 0)</parameter_value>
    <parameter_value id="param17" type="ptype4" >(1, 1, 1)</parameter_value>
    <parameter_value id="param18" type="ptype3" >20000101</parameter_value>
  </parameter values>
  <parameter sets>
    <parameter set id="pset0" >
      <parameter value="param18" />
      <parameter value="param34" />
      <parameter value="param35" />
      <parameter value="param1" />
      <parameter value="param16" />
    </parameter set>
              ⋮
  </parameter sets>
  <result values>
    <result value id="result0" type="rtype0" pset="pset3" >'result0.png'</result value>
  </result values>
  <session>
    <session result id="step0" pset="pset0" result="undefined" timestamp="0" >
      <unrelated />
    </session result>
    <session result id="step3" pset="pset3" result="result0" timestamp="1" >
      <derivation id="derivation0" >
        <calculi>
          <calculus type="application" >
            <input parameters>
            </input parameters>
            <output parameters>
              <parameter value="param19" />
              <parameter value="param2" />
            </output parameters>
          </calculus>
        </calculi>
        <input psets>
          <pset value="pset0" />
        </input psets>
        <output psets>
          <pset value="pset3" />
        </output psets>
      </derivation>
    </session result>
              ⋮
  </session>
</visualization>
```

Figure 5: The XML representation of a visualization session using the BGP visualization tool; similar portions of the document are condensed for illustration purposes. The first section (blue) details what visualization transforms were used. The next portion (green) lists all the parameter values explored, followed by the parameter sets constructed (pink) and the visualization results rendered (red). Finally, the session information is recorded (purple). This representation encodes all the information described by the model and is used to transport session information between visualization systems.

pecially those which explore the analytical uses of the model, are planned for the future.

One concern is the growth of the XML representation as visualization sessions become longer. In the worst case, the size of the file can increase quadratically with the number of results (if every new result is derived from all previous results—an unlikely case). However, in practice, the text XML does not approach anywhere near the size of the original data set for common large data sets and can be effectively compressed if needed. In the BGP example, the XML session encoding never exceeded a megabyte in size for over eighty session results. Combined with the binary PNG images for the rendered results, the overall size was 6.1 MB.

7 CONCLUSIONS

The visualization exploration process contains a wealth of information; this work has demonstrated a model to describe this information and a representation to share the information. Both the visualization technique performed and the process used to generate visualization results are captured by the model and representation.

This work impacts the user of visualization in several ways. Systems utilizing the process model assist in reuse since they clearly track where a user has been, where they are, and possibly suggest where to go. Visualizations represented using this formalism can be used in heterogeneous visualization interface environments, enabling large-scale collaboration. The salient details of the visualization process are documented, allowing others to reproduce the process. Finally, others can use the formal model to operate upon or analyze their results in a rigorous manner.

This work also contributes to the understanding of the visualization process. A characterization of user interactions with parameters during the visualization process has been performed. This characterization has led to the development of a parameter derivation calculus to describe the relationships between results created during a visualization session. Information stored using this calculus can be analyzed and further visualized to gain insight in the visualization process itself.

7.1 Future Work

This research can be extended in several ways. First, more visualization systems using this model should be developed in order to test the transport of the visualization process further. A framework is currently being constructed to assist in this task.

The parameter derivation calculus represents a wealth of information that has not been fully exploited. Different graphical visualizations and metrics based upon the calculus need to be investigated. For example, Figure 6 displays a focus+context graph visualization of the BGP visualization of Figure 4. In this example, radial distance from the center encodes how many parameter values in a result's p-set differed from the p-set of the central result; edge exist between results with only a single parameter value difference between them [12]. These sorts of visualizations may help users or designers gain insight into previous visualization sessions.

Another important aspect in many scientific visualization sessions is the change of visualization transforms. This model does not currently store any information about modifications to the transform beyond what transforms were used. A "visualization transform derivation" model is needed. However, before this could be realized, more research into unifying visualization transform representations for scientific and information visualization should be performed if any subsequent work were to be general.

Finally, the model does not currently store any meta-data. Metadata can be used to annotate any portion of the model, including results, parameter settings, or the steps in the process. A scientist's notes about a particular result or the operation performed during a state transition are all examples of meta-data to store. Meta-data would provide important semantic information about the visualization process and help capture the visualization user's insight gained from the process. Like the current model, the meta-data model needs to be flexible, allowing users to customize it to their specific application. Currently, a model similar to RDF [13] is being investigated for this purpose.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation, and the Lawrence Livermore National Laboratory. The reviewers for this paper also provided insightful input. Finally, we would like to thank the members of the UC Davis Visualization and Graphics Research Group for their input and assistance, especially Soon Tee Teoh for the use of the Border Gate Protocol visualization tool.

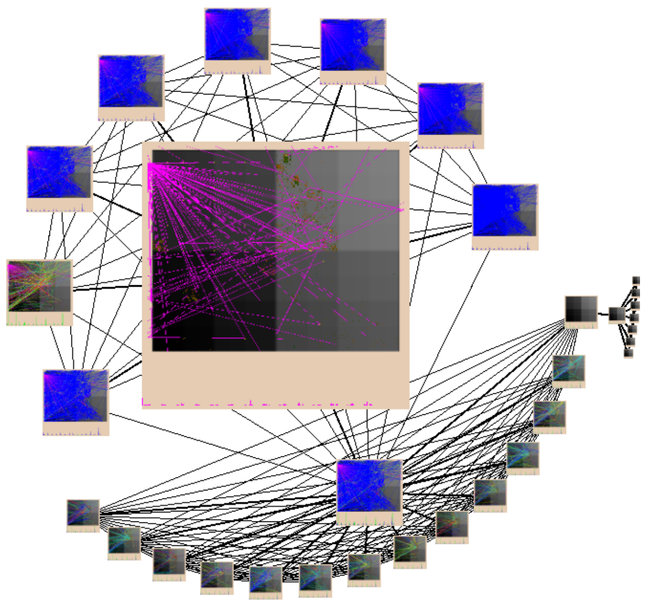


Figure 6: A focus+context visualization of the BGP visualization session in Figure 4. Results are organized radially out from the central result. The distance from the center is determined by the number of parameter differences between the central result's p-set and the p-set of the other result. Links exist between results with only one parameter value difference between them. This type of visualization can be used to get a sense of the depth of exploration of the visualization parameter space.

REFERENCES

[1] Greg Abram and Lloyd A. Treinish. An extended data-flow architecture for data analysis and visualization. *Computer Graphics*, 29(2):17–21, May 1995.

[2] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, 1967/1983.

[3] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible Markup Language (XML) 1.0 (Second Edition). Technical report, World Wide Web Consortium, 2000. http://www.w3.org/TR/REC-xml.

[4] Ken Brodlie, Andrew Poon, Helen Wright, Lesly Brankin, Greg Banecki, and Alan Gay. GRASPARC—a problem solving environment integrating computation and visualization. In *Proc. of IEEE Visualization 1993*, pages 102–109, 1993.

[5] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1999.

[6] Ed H. Chi and John T. Riedl. An operator interaction framework for visualization systems. In *Proc. the IEEE Symposium on Information Visualization 1998*, pages 63–70, 1998.

[7] James Clark. XSL Transformations (XSLT) Version 1.0. Technical report, World Wide Web Consortium, 1999. http://www.w3.org/TR/xslt.

[8] Robert B. Haber and David A. McNabb. Visualization idioms: A conceptual model for scientific visualization systems. In G.M. Nielson and B. Shriver, editors, *Visualization in Scientific Computing*. IEEE Computer Society Press, 1990.

[9] William L. Hibbard, Charles R. Dyer, and Brian E. Paul. A lattice model for data display. In *Proc. of IEEE Visualization 1994*, pages 310–317, 1994.

[10] David M. Hilbert and David F. Redmiles. Extracting usability information from user interface events. *ACM Computing Surveys*, 32(4):384–421, December 2000.

[11] T. J. Jankun-Kelly and Kwan-Liu Ma. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):275–287, July/September 2001.

[12] T.J. Jankun-Kelly and Kwan-Liu Ma. Focus+context display of the visualization exploration process. Technical report, Computer Science Department, University of California, Davis, 2002. CSE-2002-13.

[13] Ora Lassila and Ralph R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Technical report, World Wide Web Consortium, 1999. http://www.w3.org/TR/REC-rdf-syntax.

[14] C. Charles Law, Amy Henderson, and James Ahrens. An application architecture for large data visualization: A case study. In *Proc. of the 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pages 125–128, 2001.

[15] John Peter Lee. *A Systems and Process Model for Data Exploration*. PhD thesis, U. of Massachuesetts Lowell, 1998.

[16] John Peter Lee and George G. Grinstein. An architecture for retaining and analyzing visual explorations of databases. In *Proc. of IEEE Visualization '95*, pages 101–108, 1995.

[17] Kwan-Liu Ma. Image graphs - a novel approach to visual data exploration. In *Proc. of IEEE Visualization '99*, pages 81–88, 1999.

[18] Jock D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[19] Penny Rheingans. Are we there yet? Exploring with dynamic visualization. *IEEE Computer Graphics and Applications*, 22(1):6–10, January/February.

[20] Rebecca R. Springmeyer, Meera M. Blattner, and Nelson L. Max. A characterization of the scientific data analysis process. In *Proc. of IEEE Visualization '92*, pages 235–242, 1992.

[21] Soon Tee Teoh, Kwan-Liu Ma, Felix Wu, and X. Zhao. Case study: Interactive visualization for internet security. In *Proc. of IEEE Visualization '02*, 2002.

[22] Craig Upson, Thomas A. Faulhaber, Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andries van Dam. The Application Visualization System: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[23] Mark Young, Danielle Argiro, and Steven Kubica. Cantata: Visual programming environment for the Khoros system. *Computer Graphics*, 29(2):22–24, May 1995.