VisTrails: Visualization meets Data Management

Steven P. Callahan Juliana Freire Emanuele Santos
Carlos E. Scheidegger Cláudio T. Silva Huy T. Vo
SCI Institute and School of Computing – University of Utah

1. INTRODUCTION

Scientists are now faced with an incredible volume of data to analyze. To successfully analyze and validate various hypothesis, it is necessary to pose several queries, correlate disparate data, and create insightful visualizations of both the simulated processes and observed phenomena. Data exploration and visualization requires scientists to go through several steps. They need to select data sets and design complex dataflows that apply series of operations to the data to create appropriate visual representations, before they can finally view and analyze the results. Often, insight comes from comparing the results of multiple visualizations. Unfortunately, today this process contains many error-prone and time-consuming tasks. In addition, once a data product, *e.g.,* an image, is generated, all the scientist is left with is the bitmap; if a detailed caption is not created, it may not even be possible to reproduce that image at a later time. As a result, the generation and maintenance of visualizations is a major bottleneck in the scientific process, hindering both the ability to mine and use scientific data.

The VisTrails system [2, 3] represents our initial attempt to streamline the visualization process. Our long-term goal is to provide the necessary infrastructure to improve the scientific discovery process and reduce the time to insight. In VisTrails, we address the problem of visualization from a data management perspective: VisTrails *manages* the data and metadata of visualization products. By capturing the provenance of both the visualization processes and data they manipulate, VisTrails enables reproducibility and simplifies the complex problem of creating and maintaining visualizations. It also allows scientists to efficiently and effectively explore data through visualization: they can explore their visualizations by returning to previous versions of a dataflow (or visualization pipeline); apply a dataflow instance to different data; explore the parameter space of the dataflow; query the visualization history; and comparatively visualize different results. Data management techniques used in many different components of the system are key to providing these functionalities, which have been absent in previous visualization systems.

Outline. The rest of this paper is outlined as follows. In Section 2, we describe an application scenario that motivated us to build VisTrails. We discuss the limitations of existing visualization systems in Section 3 and describe the architecture of VisTrails in Section 4. Finally, we provide an overview of our demonstration in Section 5.

SIGMOD 2006, June 27–29, 2006, Chicago, Illinois, USA.

2. MOTIVATING EXAMPLE: EOFS

Paradigms for modeling and visualization of complex ecosystems are changing quickly, creating enormous opportunities for scientists and society. For instance, powerful and integrative modeling and visualization systems are at the core of Environmental Observation and Forecasting Systems (EOFS), which seek to generate and deliver quantifiably reliable information about the environment at the right time and in the right form to the right users. As they mature, EOFS are revolutionizing the way scientists share information about the environment and represent an unprecedented opportunity to break traditional information barriers between scientists and society at large [1]. However, the shift in modeling paradigms is placing EOFS modelers in an extremely challenging position, and at the risk of losing control of the quality of operational simulations. The problem results from the breaking of traditional modeling cycles: tight production schedules, dictated by real-time forecasts and multi-decade simulation databases, lead even today to tens of complex runs being produced on a daily basis, resulting in thousands to tens of thousands of associated visualization products.

As an example, Professor António Baptista, the lead investigator of the CORIE[1] project prepares figures for presentations showing results of simulations that he has designed, but that are executed by a research scientist in his group. The component elements of his figures are generated over a few hours by a sequence of scripts, activated by a different staff member in Baptista's group who is a visualization specialist. Once he receives the images, Baptista draws the composite figure for a particular run in PowerPoint using cut-and-paste. This process is repeated for similar and complementary runs. Because element components are visually optimized for each run, cross-run synthesis often have scale mismatches that make interpretation difficult.

The process followed by Baptista is both time consuming and error prone. Each of these visualizations is produced by custom-built scripts (or programs) manually constructed and maintained by several members of Baptista's staff. For instance, a CORIE visualization is often produced by running a sequence of VTK [5] and custom visualization scripts over data produced by simulations. Since there is no infrastructure to manage these scripts (and associated data), often, finding and running them are tasks that can only be performed by their creators. This is one of main reasons Baptista is not able to easily produce the visualizations he needs in the course of his explorations. Even for their creators, it is hard to keep track of the correct versions of scripts and data. Since

[1] http://www.ccalmr.ogi.edu/CORIE

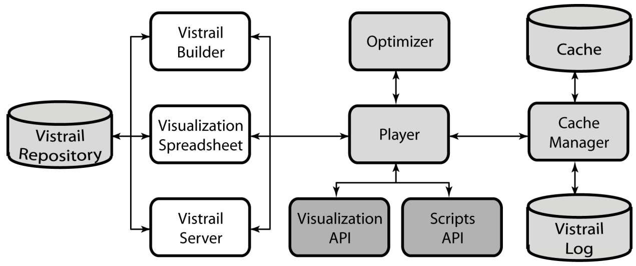


Figure 1: **VisTrails Architecture.**

these visualization products are generated in an ad-hoc manner, data provenance is not captured in a persistent way. Usually, the figure caption and legends are all the metadata available for this composite visualization in the PowerPoint slide.

3. EXISTING VISUALIZATION SYSTEMS

Visualization systems such as VTK [5] and SCIRun [6] allow the interactive creation and efficient manipulation of complex visualizations. These systems are based on the notion of dataflows, where a visualization is produced by assembling *visualization pipelines* out of modules that are connected in a network. However, these systems lack basic data management capabilities and as a result, they have important limitations.

An important limitation of existing visualization tools is that they *do not provide mechanisms for capturing provenance*. Manually created captions and filenames are often the only provenance information available for an image. They also *lack history management*—since a single instance of a dataflow is maintained, any changes to a dataflow are *destructive*. In particular, because there is no separation between the dataflow specification and its parameters, as the parameters are modified, the previous values are forgotten. This places the burden on the scientist to first construct the visualization and then to remember what values led to a particular image. As the dataflow evolves (*i.e.,* operations are added, deleted or modified) no information is kept about previous versions. Another limitation of existing tools is that they *do not provide support for comparative visualization*. In particular, they lack the necessary infrastructure for properly supporting exploratory multi-view visualizations. The process required to create and compare a large number of visualizations is way too cumbersome. For example, executing the same dataflow with different parameters (*e.g.,* different input data sets) requires users to manually specify all the parameters using a Graphical User Interface (GUI). Clearly, this mechanism is not scalable for generating more than a few visualizations. Finally, existing systems *lack an optimization infrastructure*. In particular, these systems may perform unnecessary and redundant computations while executing dataflows.

VisTrails addresses these limitations providing infrastructure to support the interaction of the scientist with the visualization process. Our objective is to give scientists a dramatically improved and simplified process to analyze and visualize large ensembles of simulations and observed phenomena.

4. THE VISTRAILS SYSTEM

The high-level architecture of VisTrails is shown in Figure 1. We only sketch the main features of the system here, for further details see [2, 3]. Users create and edit dataflows using the *Vistrail Builder* user interface. The vistrail specifications are saved in the *Vistrail Repository*. Users may also interact with saved vistrails by invoking them through the *Vistrail Server* (*e.g.,* through a Web-based interface) or by importing them into the *Visualization Spreadsheet*.

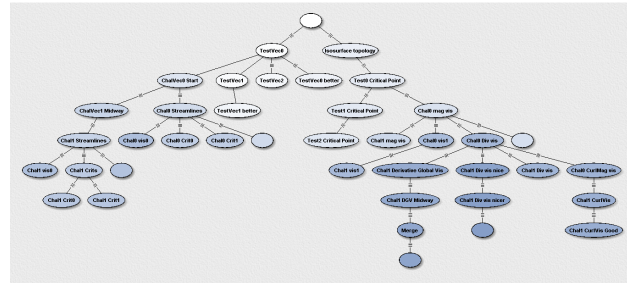


Figure 2: **A snapshot of the VisTrails history management interface. Each node in the history is a separate dataflow that differs from its parent by changes to the parameters or modules.**

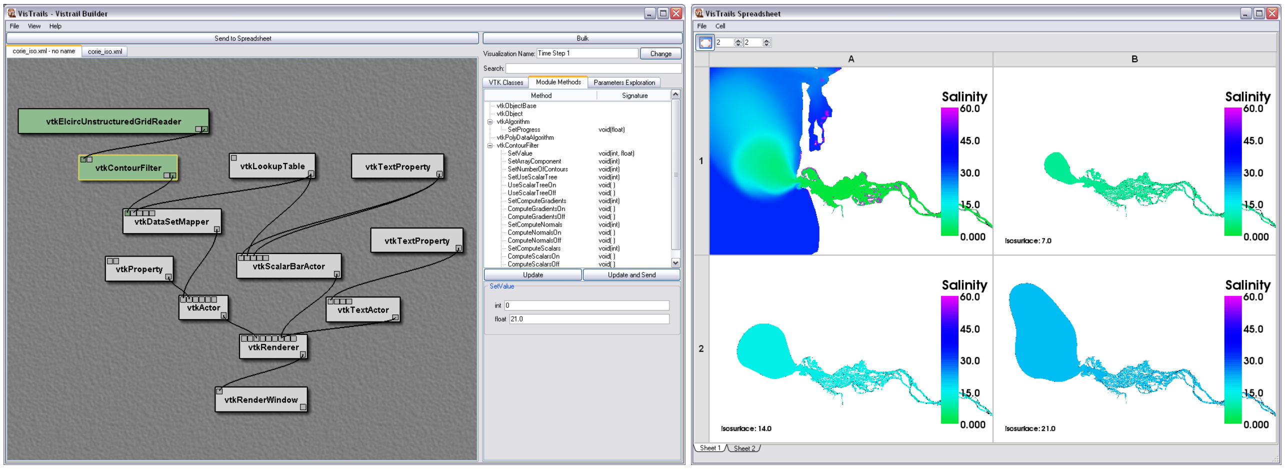
Each cell in the spreadsheet represents a view that corresponds to a dataflow instance; users can modify the parameters of a dataflow as well as synchronize parameters across different cells. Dataflow execution is controlled by the *Vistrail Cache Manager*, which keeps track of operations that are invoked and their respective parameters. Only new combinations of operations and parameters are requested from the *Vistrail Player*, which executes the operations by invoking the appropriate functions from the Visualization and Script APIs. The Player also interacts with the *Optimizer* module, which analyzes and optimizes the dataflow specifications. A log of the dataflow execution is kept in the *Vistrail Log*. The different components of the system are described below.

Vistrail Specification. A dataflow is a sequence of operations used to generate a visualization. A vistrail captures the notion of an *evolving dataflow*—it consists of several versions of a dataflow. The information in a vistrail serves both as a log of the steps followed to generate a series of visualizations, a record of the visualization provenance, and as a recipe to automatically re-generate the visualizations at a later time. The steps can be replayed exactly as they were first executed, and they can also be used as templates—they can be parameterized. In order to handle the variability in the structure of operations, and to easily support the addition of new operations, we represent vistrails using XML (for more detail, see [3]). An important benefit of using an open, self-describing, specification is the ability to query, share, and publish vistrails. This allows a scientist to locate dataflows suitable for a particular task or data products generated by a given sequence of operations, as well as to publish an image along with its associated vistrail so that others can easily reproduce the results.

History Management. As discussed above, a vistrail captures information about the evolution of a dataflow or collection of related dataflows—it behaves as a versioning system for dataflows. A vistrail consists of a tree where each node corresponds to a dataflow (see Figure 2). But instead of storing the dataflows themselves, we store the operations that take one dataflow to another. An edge between a parent and child nodes in a vistrail tree represents a set of change actions applied to the parent to obtain the dataflow for the child node. The action-based provenance mechanism of VisTrails is reminiscent of DARCS[2]. This structure it allows scientists to easily navigate through the space of dataflows created for a given exploration task. In particular, they have the ability to return to previous versions of a dataflow and compare their results. At any point in time, the scientist can choose to view the entire history of changes, or only the dataflows important enough to be given a name (*i.e.,* annotated changes).

Caching, Analysis and Optimization. Having a high-level specification allows the system to *analyze and optimize dataflows*. Ex-

[2]http://abridgegame.org/darcs



(a) (b)

Figure 3: The Vistrail Builder (a) and Vistrail Spreadsheet (b) showing the dataflow and visualization products of the CORIE data.

ecuting a vistrail can take a long time, especially if large data sets and complex visualization operations are used. It is thus important to be able to analyze the specification and identify optimization opportunities. In the current VisTrails prototype, we leverage the vistrail specification to identify and avoid redundant operations. The Vistrail Cache Manager (VCM) is responsible for scheduling the execution of modules in vistrails by identifying previously computed subnetworks and performing constant-time cache lookups.

Playing a Vistrail. The Vistrail Player (VP) receives as input an XML file for a dataflow instance and executes it using the underlying Visualization or Script APIs. The semantics of each particular execution are defined by the underlying API. Currently, the VP supports VTK classes with a very simple interpreter.

Creating and Interacting with Vistrails. The Vistrail Builder (VB) provides a graphical user interface for creating and editing dataflows (see Figure 3(a)). It writes (and also reads) dataflows in the same XML format as the other components of the system. It shares the familiar nodes-and-connections paradigm with dataflow systems. To allow users to compare the results of multiple dataflows, we built a Visualization Spreadsheet (VS). The VS provides the user a set of separate visualization windows arranged in a tabular view. This layout makes efficient use of screen space, and the row/column groupings can conceptually help the user explore the visualization parameter space [4]. The cells may execute different vistrails and they may also use different parameters for the same vistrail specification (see Figure 3(b)). To ensure efficient execution, all cells share the same cache. Users can also synchronize different cells using the VS interface.

5. DEMONSTRATION OVERVIEW

In this demonstration, we show the power and flexibility of VisTrails by presenting actual scenarios in which scientific visualization is used and showing how our system improves usability, enables reproducibility, and greatly reduces the time required to create scientific visualizations. In particular, we show how dataflows are created and modified using the Vistrail Builder and Vistrail Spreadsheet. Our examples also demonstrate the usefulness of the history management, caching capabilities, and comparative visualization tools in VisTrails.

CORIE. We demonstrate, through specific examples, how VisTrails can be used to improve the current visualization process that Professor Baptista employs. This part of the demonstration includes queries to published visualizations and the use of the history to modify existing visualizations.

Medical Imaging. Acquiring useful information from the results of medical imaging devices has been a subject of much research in the field of scientific computing. We show how VisTrails can be used to explore the parameter space using multi-view visualization and how caching substantially improves the interactivity of the process.

Uncertainty Visualization. A difficult problem in scientific simulation is to represent the uncertainty of the modeling systems due to measured or computed error. We demonstrate how VisTrails can be used to effectively visualize uncertainty through the use of evolving dataflows and comparative visualization.

An alpha release of VisTrails (available upon request) is currently being tested by a select group of domain scientists. More information about the system is available at

http://www.sci.utah.edu/~vgc/vistrails

Acknowledgments. António Baptista (Oregon Health & Science University) and Patricia Crossno (Sandia) have provided us valuable input for the system design. This work is partially supported by the National Science Foundation, the Department of Energy, Army Research Office, and IBM.

6. REFERENCES

[1] A. Baptista. *Encyclopedia of Physical Science and Technology*, chapter Environmental Observation and Forecasting Systems. Academic Press, 2002.

[2] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142, 2005.

[3] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Managing the evolution of dataflows with vistrails. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow 2006)*, 2006. To appear.

[4] E. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *IEEE Information Visualization Symposium*, pages 17–24, 1997.

[5] Kitware. The Visualization Toolkit (VTK) and Paraview. http://www.kitware.com.

[6] S. G. Parker and C. R. Johnson. SCIRun: a scientific programming environment for computational steering. In *Supercomputing*, 1995.