Provenance in Comparative Analysis

A Study in Cosmology

Provenance—the logging of information about how data came into being and how it was processed—is an essential aspect of managing large-scale simulation and data-intensive projects. Using a cosmology code comparison project as an example, this article presents how a provenance system can play a key role in such applications.

The continuing growth of computational power and algorithmic techniques is leading to the adoption of an increasingly varied array of simulations and processing methodologies for scientific discovery. The size of simulation and observational databases is growing rapidly as well, posing major challenges in the domain of data analysis and database queries.[1] This increase in both data size and analysis modalities makes tracking the history—or *provenance*—of final results as well as intermediate data products more important than ever. Provenance in this context includes not only the data produced but the exact processes and parameters required to reproduce them.

Attempting to collect and manage large data sets by hand is impossible given the computationally intensive and dynamic nature of modern scientific data analysis and exploration. Similar situations have arisen in software development (because of version control), large-scale numerical simulations (because of parameter dependence), and observational data sets (because of historical information). Examples from astronomy include the Flexible Image Transport System (FITS)[2] and the Image Reduction and Analysis Facility (IRAF),[3] both of which maintain detailed information about data's origin and processing. The need for a flexible methodology capable of automatically tracking and managing the provenance of an entire analytical process is thus widely recognized.[4]

Cosmology presents an excellent example of a field with both large-scale simulations and data sets. Many next-generation cosmological observations are targeted at remarkably high accuracy levels (on the order of 1 percent) and are increasingly rich in terms of spectral and spatial coverage. A key task for cosmological theory is to keep pace with—or preferably exceed—the demanding requirements set by observations. Consequently, researchers have developed many different simulation codes to model various aspects of the universe, ranging from predicting the distribution of matter on the largest observable scales to peering into the formation of individual objects. Because each code is different, in both its approach to modeling the underlying data and the method by which it controls numerical error, it requires a unique parameterization to generate the proper

1521-9615/08/$25.00 © 2008 IEEE
Copublished by the IEEE CS and the AIP

Erik W. Anderson and Cláudio T. Silva
University of Utah
James P. Ahrens, Katrin Heitmann, and Salman Habib
Los Alamos National Laboratory

result. Thus, we must consider a given result's robustness as a function of the individual codes that produced it: if all the codes are in close agreement, our confidence in the final result is significantly enhanced.

Code comparisons that involve multiple teams have become an important part of establishing a result's robustness.[5] The process of making complicated comparisons across multiple codes, which sometimes includes the development of new codes (or different versions of the same code), underscores the need for provenance-aware tools. The example presented in this article shows how we addressed some of these challenges by using a particular provenance management system (see the "Cosmic Code Comparison Project" sidebar).

What Is Provenance?

Provenance isn't limited to passively tracking a single datum:[6,7] it also records the *process* that governed the data's creation or manipulation. The VisTrails provenance management system[8,9] represents the process operating on given data as a workflow. Figure 1 shows an example that renders a simulation's particles as cone glyphs. The named version of this process in the VisTrails version tree, *Halo-Flash particles*, appears along with the resulting visualization as insets in the figure. In many cases, examining the method used to generate or manipulate a datum is more useful than only taking the data products into account.

By capturing changes to the process by which we create or modify data, the VisTrails system can present a concise view of process provenance, and by combining VisTrails with a database, we can make a relation between any data and the process that generated it. We can then query these relations to get even more information and gather additional insights. Figure 2 shows an overview of some of the VisTrails system's components. Each node in the version tree represents a complete workflow and is responsible for creating a data product, be it a visualization, a file, or some other report. Each node also has textual notation attachments to let users record their thoughts and insights during each step of the analysis. Because the visualizations that each workflow produces are viewable in the visualization spreadsheet (Figure 1b), users can compare both the data and the impact of different processes applied to that data.

Provenance doesn't merely provide an ideal mechanism to reproduce results—it also lets scientists fully share their experiments and analysis techniques with other domain experts. Researchers can confidently apply a guaranteed process to other data because the entire processing pipeline is fully described along with all the parameters necessary to duplicate an experiment.

The aim of adding workflow management and provenance tracking to a code comparison project is to let users more quickly and accurately analyze

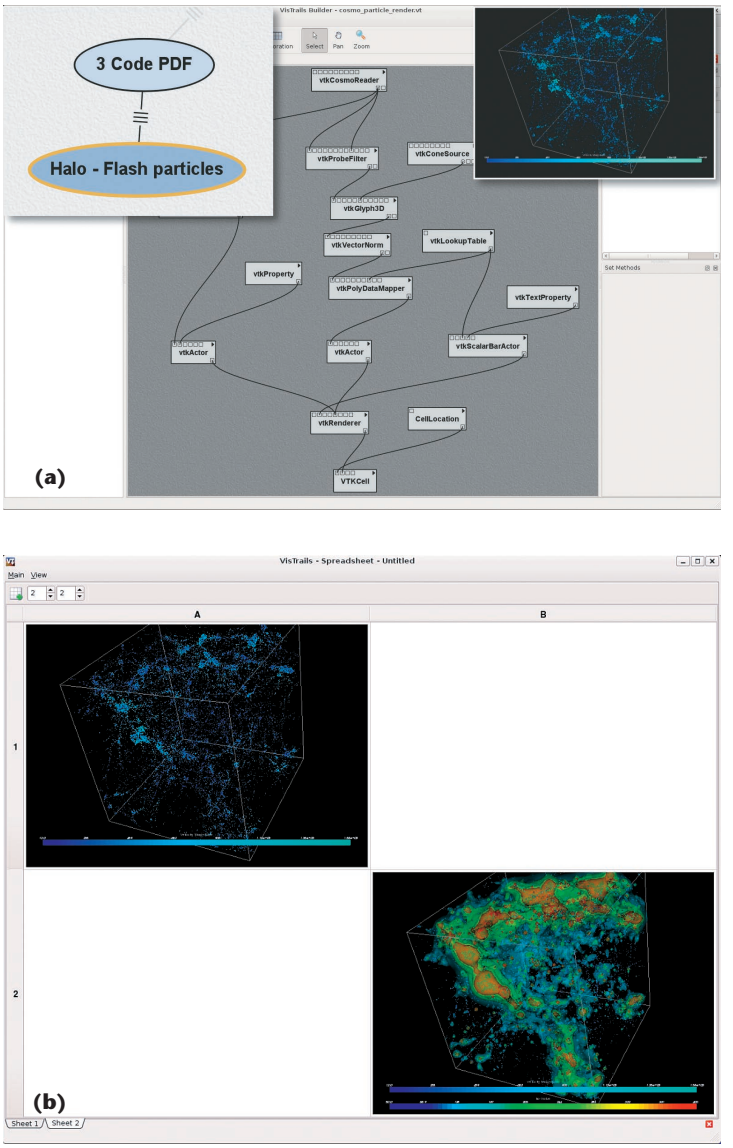

Figure 1. Workflow overview. (a) VisTrails represents a process operating on data as a workflow in which several atomic operations form a processing algorithm. The insets show this workflow's resulting visualization: a rendering of a cosmological simulation in which cone-shaped glyphs represent the particles. The version containing this workflow is a single element of a larger version tree, highlighted in orange. (b) Next to the workflow description is the visualization spreadsheet, which is used to manage and display multiple visualizations simultaneously.

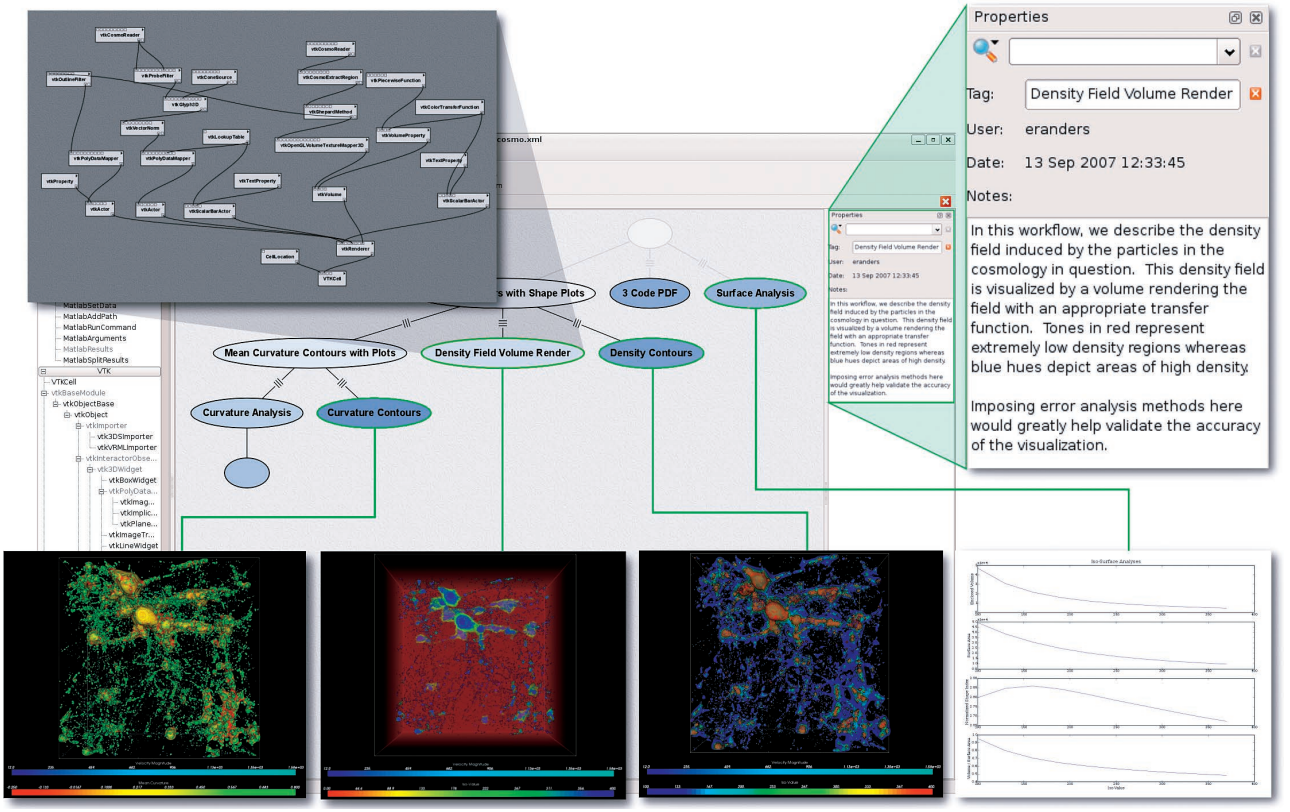

Figure 2. VisTrails system components. VisTrails captures and manages the complete provenance of all explorations performed on the data. Here, the provenance information is displayed as a history tree, with each node representing a workflow that generates a unique visualization. Each VisTrail version node also stores detailed metadata, including free-text notes, the date and time the workflow was created or modified, optional descriptive tags, and the user who created it.

The Cosmic Code Comparison Project

Researchers can't always perform controlled physical experiments in fields such as cosmology and astrophysics. Consequently, the reliability of simulations of multiscale, nonlinear processes (and their applicability domain) comes attached with a question mark. Simulation result robustness is particularly open to scrutiny.

The Cosmic Code Comparison Project[1] at Los Alamos National Laboratory aims to establish the robustness of cosmological simulation results for a chosen set of applications relevant to precision near-term observational campaigns. It also aims to analyze the strengths and weaknesses of various simulation codes as they relate to each other (although a systematic code comparison is complicated by different simulations and the underlying algorithms around which they're built). So far, researchers have compared the results from 10 different simulation codes over numerous tests. Although they're sometimes idealized, these tests have already helped identify shortcomings and quirks in various simulation techniques. Clearly, this type of study is important not just in cosmology but in many other application arenas as well.

Reference

1. K. Heitmann et al., *The Cosmic Code Comparison Project*, Computational Science and Discovery, 2007.

and describe the differences between codes. An extra benefit is the enhanced ability to not only run an analysis on a given simulation but also to ensure that we keep a useful record of the exact parameterization governing it. This information includes the parameters required for each processing step as well as the data used as input for the analysis.

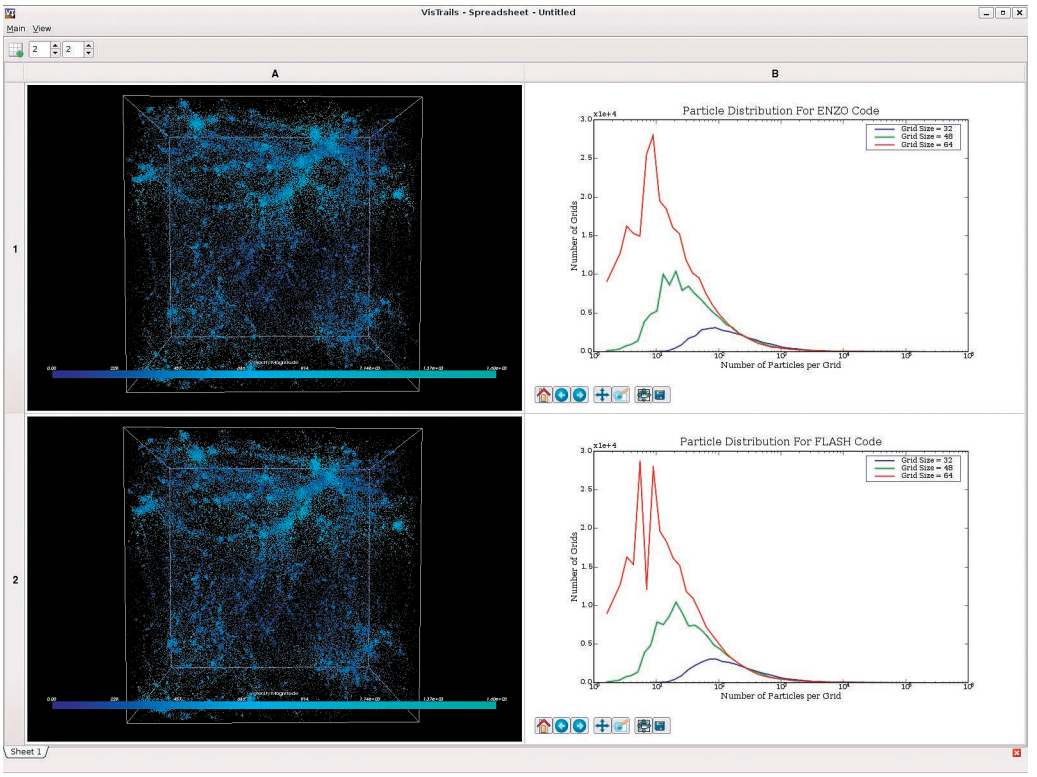

Figure 3. Side-by-side comparison of two different simulation codes. The particle visualization in the top row, representing the results of the ENZO code, looks identical to the particle visualization of the FLASH code on the bottom row. With additional quantitative information, such as a histogram of particle density measured at various grid sizes, we can spot differences in the two codes' results.

Provenance for Data Comparison

Researchers often use visualizations to explore cosmological data and analysis results.[10,11] Visualization of a cosmological data set, such as a simulation code's result, can clearly convey the data's structure, but visualizations also depend on the input parameters to both the simulation code and the visualization algorithm itself.

One common method for examining the differences between parameterizations is *comparative visualization*. We start by laying out side-by-side visualizations in a spreadsheet of elements; by synchronizing the interaction between the various cells, we can ensure a comparison between the same regions in each rendering. Presenting a group of visualizations together is important, but using the management capabilities of a unified display mechanism such as a spreadsheet helps us synchronize visualizations, which is useful if we want to compare several of them.

An important impact of comparative visualization is the ability to determine exactly how two related images differ and how they're produced: by inspecting the provenance associated with each rendering, we can start answering questions about the steps taken to produce the two visualizations. Because VisTrails maintains a complete record of the steps taken to modify and process the data, we can examine individual actions and determine how each one affects the perception of the data on display. Such actions also include interaction with the data: users can more easily compare multiple visualizations and determine the best data representations by applying specific camera positions, for example, which are then stored as part of the visualization's provenance. Figure 3 shows how applying provenance in this way to multiple visualizations can produce a spreadsheet populated with the related, aligned visualizations necessary for a concrete understanding of a data product's structure.

Structural Comparisons

Useful processing techniques for cosmologists, such as determination of the gravitationally bound objects known as halos, are often represented as a series of computational modules, each operating on the data. Figure 4 represents this *workflow*

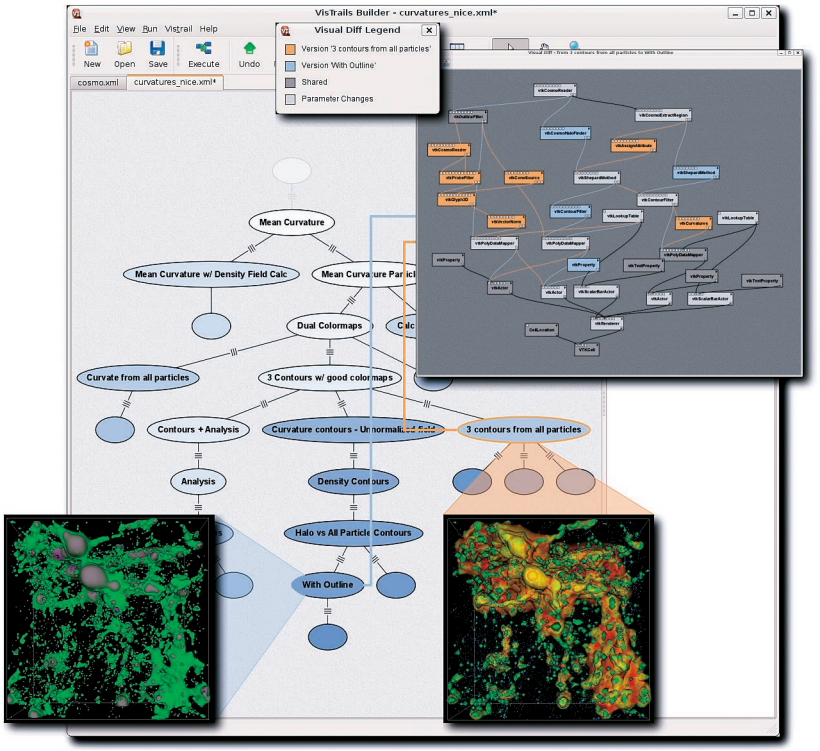

Figure 4. Workflow description. By representing provenance as a series of actions that modify a pipeline, we can visualize the differences between two workflows, which reduces the time it takes us to understand how those two workflows functionally differ.

description (or computational structure) as a collection of specifically connected modules. Understanding a workflow's structure and how it relates to the data flowing through it offers us another way to investigate simulation results.

Well-made visualizations and renderings can convey large quantities of data in a clear and easily understood manner. However, any good visualization matches processing and analysis techniques with visualization and rendering algorithms, and each workflow step contributes its own set of parameters to the overall parameter space governing the rendering's final outcome. The problem of trying to examine and compare one large workflow with another by exploring each one individually becomes more difficult as the task's complexity increases. The interesting—and arguably most important—aspects of two such workflows lie not in their complete descriptions, but in their differences. Figure 4, for example, depicts two similar workflows that yield drastically different results, but by analyzing the differences between them, we can easily explore each visualization's salient aspects.

One method of analyzing two separate workflows is to use each one's provenance information, much like the VisTrails system does. Although the *visual diff* shown in Figure 4 displays only the difference between two well-made visualizations, we can still examine the explorations used to form these two products more thoroughly—specifically, we can gather a more complete view of the thought processes behind the two visualizations by forming visual diffs incrementally. These diffs will show the step-by-step creation of each product, which is a very difficult task without careful provenance management.

Collaborative Environments

In collaboratories, many scientists might work on the same data and a common collection of workflows—seamlessly integrating input and explorations from different team members, regardless of their expertise, is an important and challenging task that can lead to new ideas and insights.

Collaboration between many researchers means not only sharing input data and results but also detailed metadata about the experiments performed. Figures 2 and 4 depict representation and interaction with provenance metadata. Giving users the ability to ask questions about workflows, whether other users created them or not, provides another tool for more thorough data exploration. Using the visual diff functionality between pipelines created by other users can help them more concretely understand another person's thought process. This type of metadata exploration reinforces the ideas of experimental reproducibility and verification.

Visual diffs aren't the only provenance-based mechanism at the cosmologists' disposal. Figure 2 shows additional metadata in the form of written text associated with each node in the version tree. Because these free-text notes are part of the provenance information stored with each version, collaborating users are free to record proper usage of a processing technique, the motivation for applying a certain visualization algorithm to their data, the reasoning behind specific parameter values, or anything else that might affect the workflow being considered.

Reproducibility

The concept of reproducibility is intrinsic to the

scientific method, chiefly, the notion that deterministic code bases (that is, those with no intrinsic stochastic properties) coupled with known sets of parameter settings (including historical information) are sufficient to ensure exactly repeatable results. The application of visualization and analysis tools involves another set of parametric (and other) choices, including tracking the various tools applied. The history of these settings and choices is thus an equally important part of reproducibility, especially in complex simulation and observational databases.

Learning by Example

The "learn by doing" method of exploration is often the quickest way to become familiar with a processing technique. It's common for beginners to ask questions about the exact steps an expert took to generate a specific data product, but answering these questions can easily consume significant amounts of time.

We can largely avoid this problem by integrating provenance information into day-to-day research activities. Because provenance is, by definition, never destroyed by a management system, a complete record of data exploration exists. By combining the complete exploration history, along with relevant notes from experts along the way (see Figure 2), beginners can often find answers to their questions without having to ask them directly.

Moreover, by simply making a copy of the provenance information associated with a given exploration, we can not only follow the expert's exploration of the data but also learn how to extend it. Such individual explorations can then lead to a more complete understanding of the data and the processing techniques used to generate meaningful results. Progressing beyond the initial familiarization phase about data and the processing techniques used on it means we can ask more insightful questions, which leads to more meaningful interactions with experts.

Figure Reproducibility

Numerical results in published research papers are presented statically, most often in the form of figures. However, reproducing figures—and being able to dynamically explore the processes that created them—is an important part of understanding the underlying research. Distributing the workflows that describe these processes is thus a first step toward guaranteeing reproducibility (AMRITA [www.amrita-cfd.org] takes a more immersive approach).

Including provenance information with a paper's images and figures lets anyone who reads it recreate the exploration in its entirety. The level of reproducibility possible using a provenance-aware mechanism far exceeds that of methodologies not taking advantage of history tracking.

Reproducing and Reapplying Analyses

Although published processing techniques are carefully described in the literature, implementing the algorithms behind them is rarely a trivial task. Fortunately, many complicated algorithms are available through the use of specialized libraries.

Many of these libraries can implement complex algorithms and processing techniques, but their use is often highly sensitive to the given parameters. To properly apply a processing technique, we must know the intricacies about both the data being processed and the technique being used. Provenance can help researchers apply techniques developed by others to explore the different parameterizations used to generate a known result, thereby easing the learning curve associated with the techniques. Without provenance information attached to the workflows describing the use of these techniques, it's often much harder to determine the method's proper application to different data.

Many, if not most, systems provide some degree of transparent history storage—for example, Matlab provides history in the form of commands issued to the interpreter. Although this is a very powerful tool for users, it doesn't provide a complete view of the provenance associated with a specific result: the provenance in this case exists only as a list of commands used to generate the last data product computed. Reproducing intermediate or multiple results is difficult at best, but we can ease this difficulty by associating the complete set of steps, or workflow, with the proper data product. In this way, simple execution histories become more general and useful. In our work, we use VisTrails to manage our provenance because it already contains a comprehensive set of provenance and workflow features. Although the VisTrails model has proven adept at tracking the processing pipeline's provenance, work must still be done to provide better access to data stored in databases. This additional component could bridge gaps between the pipeline's provenance and that of a given data product.

Enabling Provenance in Existing Tools

Not every tool used to process simulation results was created with provenance in mind, which is preventing the scientific community from widely adopting provenance-tracking frameworks.

VisTrails provides a simple way to add user-defined modules to a system. A VisTrails *package* is a collection of Python classes that expose existing code for use in workflows. If the code is already accessible inside Python (many high-quality scientific libraries already exist in the language), it typically takes just a few minutes to expose the code as modules in VisTrails. If the code is only available in C++ or Fortran libraries, existing software packages can simplify the creation of *foreign function interfaces*, which are simply ways to call functions across programming languages. If neither of these options apply, Python's extensive system capabilities can execute the external processes.

A comprehensive guide to writing VisTrails packages is available at www.vistrails.org; Figure A shows some example code for a VisTrails module.

```
import my_library
import core.modules
import core.modules.module_registry
from core.modules.vistrails_module import Module, ModuleError

version = '0.1'
name = 'MyLibrary'
identifier = 'com.my_domain.my_library'

class MyFunc(Module):
  """ This module calls the same function from within VisTrails """
  def compute(self):
   input_value = self.getInputFromPort("Input0")
   result = my_library.my_function(input_value)
   self.setResult("Output", result)

def initialize(*args, **keywords):
  reg = core.modules.module_registry
  basic = core.modules.basic_modules

  reg.add_module(MyFunc, name="Display Name")
  reg.add_input_port(MyFunc, "Input0", (basic.Float, 'Input
Description'))
  reg.add_output_port(MyFunc, "Output", (basic.String, 'Output
Description'))
```

Figure A. VisTrails wrapper for an external Python-accessible library function. This small amount of code exposes an external library function for use within the VisTrails system.

Acknowledgments

This work was partially supported by the US National Science Foundation, the US Department of Energy, and IBM faculty awards.

References

1. I.T. Foster and C. Kesselman, "Scaling System-Level Science: Scientific Exploration and IT Implications," *Computer*, vol. 39, no. 11, 2006, pp. 31–39.
2. D.C. Wells, E.W. Greisen, and R.H. Harten, "FITS: A Flexible Image Transport System," *Astronomy and Physics Supplement*, vol. 44, June 1981, p. 363.
3. D. Tody, "The IRAF Data Reduction and Analysis System," *Instrumentation in Astronomy VI*, Soc. Photo-Optical Instrumentation Engineers, vol. 627, 1986, pp. 733–753.
4. C.T. Silva, J. Freire, and S.P. Callahan, "Provenance for Visualizations: Reproducibility and Beyond," *Computing in Science & Eng.*, vol. 9, no. 5, 2007, pp. 82–89.
5. K. Heitmann et al., *The Cosmic Code Comparison Project*, Computational Science and Discovery, 2007.
6. P. Missier et al., "Requirements and Services for Metadata Management," *IEEE Internet Computing*, vol. 11, no. 5, 2007, pp. 17–25.
7. L. Moreau and I.T. Foster, "Provenance and Annotation of Data," *Proc. Int'l Provenance and Annotation Workshop*, LNCS 4145, Springer-Verlag, 2006.
8. L. Bavoil et al., "VisTrails: Enabling Interactive Multiple-View Visualizations," *Proc. IEEE Visualization*, IEEE Press, 2005, pp. 135–142.
9. J. Freire et al., "Managing Rapidly Evolving Scientific Workflows," *Proc. Int'l Provenance and Annotation Workshop*, LNCS 4145, Springer-Verlag, 2006, pp. 10–18.
10. C.D. Hansen and C. Johnson, *Visualization Handbook*, Academic Press, 2004.
11. W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit*, 3rd ed., Kitware, 2004.

Erik W. Anderson *is a research assistant and PhD candidate at the University of Utah. His research interests include scientific visualization, signal processing, computer graphics, and multimodal visualization. Anderson has a BS in computer science and a BS in electrical and computer engineering from Northeastern University. Contact him at eranders@sci.utah.edu.*

James P. Ahrens *is a technical staff member at Los*

Alamos National Laboratory. His research interests include visualization and high-performance computing. Ahrens has a PhD in computer science from the University of Washington. He is a member of the IEEE Computer Society. Contact him at ahrens@lanl.gov.

Katrin Heitmann *is a technical staff member at Los Alamos National Laboratory. Her research focuses on computational cosmology, especially on simulating the large-scale structures observed in the universe. Heitmann has a PhD in physics from the University of Dortmund, Germany. She is a member of the American Physical Society. Contact her at heitmann@lanl.gov.*

Salman Habib *is a technical staff member at Los Alamos National Laboratory. His research interests include dynamics—nonlinear and stochastic—of classical and quantum field theories and applications in particle physics, condensed matter systems, atomic and quantum optics, accelerator physics, and cosmology. Habib has a PhD in physics from the University of Maryland, College Park. He is a member of the American Physical Society. Contact him at habib@lanl.gov.*

Cláudio T. Silva *is an associate professor at the University of Utah. His research interests include visualization, geometry processing, graphics, and high-performance computing. Silva has a PhD in computer science from SUNY at Stony Brook. He is a member of the IEEE, the ACM, Eurographics, and Sociedade Brasileira de Matematica. Contact him at csilva@cs.utah.edu.*