

Simplifying the Design of Workflows for Large-Scale Data Exploration and Visualization

Juliana Freire Cláudio T. Silva
{juliana,csilva}@cs.utah.edu
University of Utah

Introduction

Computing has been an enormous accelerator to science and has led to an information explosion in many different fields. To analyze and understand scientific data, complex computational processes must be assembled, often requiring the combination of loosely-coupled resources, specialized libraries, and grid and Web services. Workflow systems have therefore grown in popularity within the scientific community [2]. Not only do workflows support the automation of repetitive tasks, but they can also capture complex analysis processes at various levels of detail and systematically capture provenance information for the derived data products [1, 3]. The provenance (also referred to as the audit trail, lineage, and pedigree) of a data product contains information about the process and data used to derive the data product. It provides important documentation that is key to preserving the data, to determining the data’s quality and authorship, and to reproduce as well as validate the results. These are all important requirements of the scientific process.

But applying traditional workflow technology to exploratory tasks brings new challenges. Whereas business workflows are primarily used to automate repetitive processes, scientific workflows are often used in exploratory tasks, where change is the norm. Furthermore, scientific workflows need to cater to a broader set of users, including many who do not have programming expertise. Even for systems that have sophisticated visual programming interfaces, the path from the raw data to insights is quite laborious and error-prone. Visual programming interfaces expose computational components (functions) as *modules* and allow the creation of complex pipelines¹ which combine these modules in a workflow, where data flows along the *connections* between modules. These interfaces ease the creation of pipelines through the use of a simple programming model (dataflows) and by providing built-in constraint checking mechanisms (e.g., that disallow a connection between incompatible module ports).

Notwithstanding, without detailed knowledge of the underlying computational components, it is difficult to understand what series of modules and connections ought to be added to obtain a desired result. In essence, there is no “roadmap”: systems provide very little feedback to help the user figure out which modules can or should be added to the pipeline. This problem is compounded by the fact that analysis pipelines contain many disparate components: data needs to be gathered, generated, integrated, transformed, and visualized. A novice user, or even an advanced user performing a new task, often resorts to manually searching for existing pipelines to use as an example. These examples are then adapted and iteratively refined until a solution is found. Unfortunately, this manual, time-consuming process is the current standard for creating pipelines rather than the exception.

In this paper, we describe an infrastructure we developed to support the design of large-scale exploratory computational tasks, and in particular of data analysis through visualization. The infrastructure includes a set of *scalable tools and intuitive interfaces that allow casual users to explore and re-use the knowledge embedded in pipeline specifications*. By querying the task specifications, users can learn by example from the reasoning and/or analysis strategies of experts; expedite their training; and potentially reduce the time lag between data acquisition and insight. We also discuss how information in collections of pipeline specifications can be explored simplify the creation and refinement of pipelines. In particular, we show how this information can be used to build a recommendation system and a mechanism that for modifying pipelines by analogy.

In our presentation, we will give a live demonstration of this infrastructure that has been implemented in the Vis-Trails system (<http://www.vistrails.org>). We will also discuss current efforts to use this infrastructure in a workflow collaboratory that supports social analysis of scientific data [4].

¹We use the terms pipelines and workflows interchangeably.

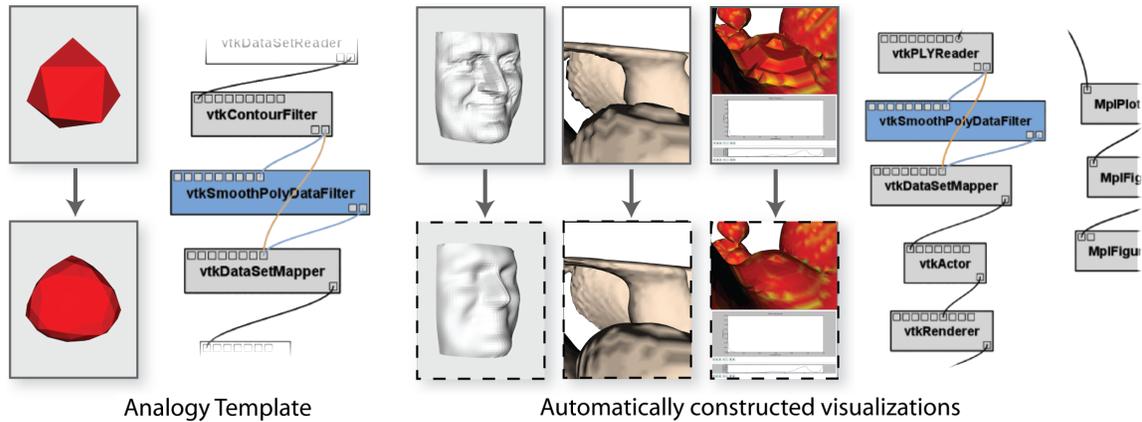


Figure 1: Workflow refinement by analogy. The user chooses a pair of workflows to serve as an analogy template. In this case, the pair represents a change where a file downloaded from the WWW is smoothed. Then, the user chooses a set of other workflows that will be used to derive new visualizations, with the same change. These new visualizations are derived automatically. The workflow on the left reflects the original changes, and the one on the right reflects the changes when translated to the last visualization on the right. The workflow pieces to be removed are portrayed in orange, and the ones to be added, in blue. Note that the surrounding modules do not match exactly: the system determines the most likely match.

Querying and Modifying Workflows by Analogy

The ability to query workflow specifications enables users to better understand exploratory processes and their results, as well as re-use knowledge. For example, a user can identify workflows (or subworkflows) that are suitable for a given task. However, querying workflows using traditional text-based interfaces (e.g., using languages such as SQL) is cumbersome. Because workflows are represented as graphs, to specify a query over the workflow structure effectively requires that a subgraph query be encoded as text. We have developed a query-by-example (QBE) interface that allows users to quickly construct expressive queries using the same familiar interface they use to build workflows [6]. This allows users, who may not be familiar with specialized query languages, to effectively explore the information in a workflow repository.

Even starting from examples, the process of assembling workflows requires significant effort. It also requires programming expertise as well as detailed knowledge of multiple tools and libraries. Although many scientific workflow systems provide sophisticated visual programming interfaces, the path from the raw data to insights is still laborious and error-prone. To address this problem, we have designed a new mechanism for creating and refining workflows by analogy [6]. While the query interface allows users to identify workflows (and sub-workflows) that are relevant for a particular task, the *analogy operation* provides a mechanism for reusing these pipelines to construct new data products in a semi-automated manner—without requiring users to directly manipulate or edit the workflow specifications. Figure 1 illustrates the use of analogies for creating visualizations. To apply an analogy, first the difference between a source pair of analogous visualizations is computed. This difference is then transferred to a third visualization. Analogies can be used as the basis for scalable updates: the user does not need to have knowledge of the exact details of the three visualization workflows to perform the operation.

A Recommendation System for Workflow Design

Analogies are a powerful mechanism and can be helpful in situations when the user knows in advance what they want the end result to be. However, during workflow creation, it is not always the case that the user has an analogy template readily available for the result that is desired. In these cases, the user is relegated to manually searching for examples.

We have developed a recommendation system for guiding users through the process of creating workflows [5]. By identifying common paths used in existing workflows, the recommendation system predicts a set of likely module sequences that can be presented to the user as suggestions during the design process. The system acts as an auto-completion mechanism for workflows, suggesting potential modules and connections in a manner similar to a Web browser suggesting URLs. The workflow completions are presented graphically in a way that allows the user to easily explore and accept suggestions or disregard them and continue working as they were. Figure 2 gives an overview of how the auto-completion mechanism works.

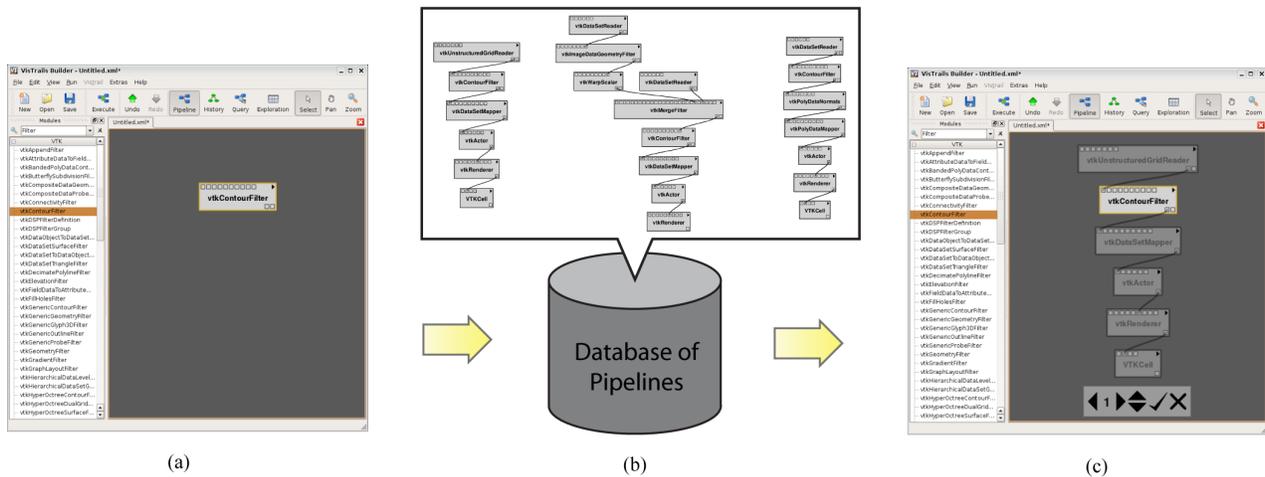


Figure 2: Workflow auto completion. (a) A user starts by adding a module to the pipeline. (b) The most likely completions are generated using indexed paths computed from a database of pipelines. (c) A suggested completion is presented to the user as semi-transparent modules and connections. The user can browse through suggestions using the interface and choose to accept or reject the completion.

Acknowledgments

We thank the members of the VisTrails group, our students, and collaborators for their contributions to this work. Our research has been funded the Department of Energy SciDAC (VACET and SDM centers), the National Science Foundation (grants IIS-0746500, CNS-0751152, IIS-0713637, OCE-0424602, IIS-0534628, CNS-0514485, IIS-0513692, CNS-0524096, CCF-0401498, OISE-0405402, CCF-0528201), and IBM Faculty Awards (2005, 2006, 2007, and 2008).

References

- [1] S. B. Davidson, S. C. Boulakia, A. Eyal, B. Ludäscher, T. M. McPhillips, S. Bowers, M. K. Anand, and J. Freire. Provenance in scientific workflow systems. *IEEE Data Eng. Bull.*, 30(4):44–50, 2007.
- [2] E. Deelman and Y. Gil. NSF Workshop on Challenges of Scientific Workflows. Technical report, NSF, 2006. http://vtcpc.isi.edu/wiki/index.php/Main_Page.
- [3] J. Freire, D. Koop, E. Santos, and C. T. Silva. Provenance for computational tasks: A survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.
- [4] J. Freire and C. Silva. Towards enabling social analysis of scientific data. In *CHI Social Data Analysis Workshop*, 2008. To appear.
- [5] D. Koop, C. Scheidegger, S. Callahan, J. Freire, and C. Silva. Viscomplete: Data-driven suggestions for visualization systems. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008. Papers from the IEEE Visualization Conference 2008.
- [6] C. Scheidegger, D. Koop, H. Vo, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, 2007. Papers from the IEEE Visualization Conference 2007.