Managing Exploratory Workflows

Juliana Freire
Claudio T. Silva
http://www.sci.utah.edu/~vgc/vistrails/
University of Utah

Joint work with:
Erik Andersen, Steven P. Callahan, David Koop,
Emanuele Santos, Carlos E. Scheidegger and Huy T. Vo
Workflows and Scientific Discovery

- Workflows are emerging as a paradigm for representing and managing complex computations
- They capture computation and analysis processes, enabling
  - Automation
  - Reproducibility
  - Result sharing
- Potential to accelerate and transform the scientific analysis process
- But... existing systems fail to provide the necessary infrastructure for exploratory tasks
Exploration and Workflows

- Workflows have been traditionally used to automate repetitive tasks
- In exploratory tasks, *change is the norm!*
  - Data analysis and exploration is an iterative process

Figure modified from J. van Wijk, IEEE Vis 2005
Data Exploration and Workflows: Today

raw data

---

Files

- anon4877_voxel_scale_1_zspace_20060331.srn
- anon4877_textureshading_20060331.srn
- anon4877_textureshading_plane0_20060331.srn
- anon4877_goodxferfunction_20060331.srn
- anon4877_lesion_20060331.srn

---

Notes

- Initial visualization with z-scaling corrected
- Added texture and shading
- Added plane to visualize internal structure
- Found good transfer function
- Identified lesion tissue
Data Exploration and Workflows: Issues

- Data provenance is maintained manually through file-naming conventions and detailed notes
  - A time-consuming process
- Hard to understand the exploratory process and relationships among workflows
- Hard to further explore the data, e.g., locate relevant data products/workflows and modify them
- Hard to collaborate
  - Work is likely to be lost if creator leaves

*The generation and maintenance of workflows is a major bottleneck in the scientific process*
Need Support for Reflective Reasoning

- Reflective reasoning is key in the scientific process
- “Reflective reasoning requires the ability to store temporary results, to make inferences from stored knowledge, and to follow chains of reasoning backward and forward, sometimes backtracking when a promising line of thought proves to be unfruitful. ...the process is slow and laborious”
  
  Donald A. Norman

- Need external aids—tools to facilitate this process
- Need aid from people—collaboration

Need Data Management!
VisTrails: Managing Exploration

- Streamlines the creation, execution and sharing of complex data products

- VisTrails manages the data, metadata and the exploration process, scientists can focus on science!

- Not a replacement for visualization or scientific workflow systems: provides infrastructure that can be combined with and enhance these systems

- Focus on usability—build tools for scientists
Demo: Action-Based Provenance and Workflow Evolution

Action-Based Provenance

- Records changes applied to workflows
- Workflow evolution is captured in a *vistrail*—a rooted tree where
  - *nodes* correspond to workflow versions
  - *edges* correspond to actions that transform the parent into the child workflow
- Action algebra:
  - addModule, deleteModule, addConnection, deleteConnection, setParameter, ...
  - Can be easily extended, e.g., addDirector for Ptolemy-based systems
Action-Based Provenance: Example

- addModule
- deleteConnection
- addConnection
- addConnection
- setParameter
Let

- $DF$ be the set of all possible dataflow instances, s.t. $\emptyset \in DF$
- $x_i: DF \rightarrow DF$ be a function that transforms a dataflow $x_i(D_a) = D_b$

A vistrail node $v_t$ corresponds to the dataflow that is constructed by the sequence of actions from the root to $v_t$

$$v_t = x_n \circ x_{n-1} \circ \ldots \circ x_1 \circ \emptyset$$

$$\text{decimate} = x_3 \circ x_2 \circ x_1 \circ \emptyset$$
Action-Based Provenance: Summary

- Uniformly captures both data and process provenance
- Records user actions—compact representation
- Detailed information about the exploration process
  - Results can be reproduced
- Provenance beyond reproducibility:
  - Scientists can return to any point in the exploration space
  - Enables scalable exploration of the parameter space
    (and compare results using spreadsheet!)
  - Simplifies collaboration
  - Understand problem-solving strategies—knowledge re-use
Demo: Bulk Updates and Parameter-Space Exploration
Bulk Updates

Generating Animations

Scalable Derivation of Data Products

- Scripting workflows: Bulk updates are simple to specify and apply
- Exploration of parameter space for a workflow $\mathbf{v}_t$
  
  $(\text{setParameter}(id_n, value_n) \circ \ldots \circ (\text{setParameter}(id_1, value_1) \circ \mathbf{v}_t))$

- Exploration of multiple workflow specifications
  
  $(\text{addModule}(id_i, \ldots) \circ (\text{deleteModule}(id_i) \circ \mathbf{v}_1) \circ \ldots \circ (\text{addModule}(id_i, \ldots) \circ (\text{deleteModule}(id_i) \circ \mathbf{v}_n))$

- Results can be conveniently compared in the VisTrails spreadsheet
- Can create animations too!
- Caching to avoid redundant computations
Interacting with Provenance

- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level
Interacting with Provenance

- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level
- Create views over the version tree
Interacting with Provenance

- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level
- Create *views* over the version tree
Three Layers of Metadata

Workflow Evolution

Workflow

Execution

<table>
<thead>
<tr>
<th>workflow_exec_id</th>
<th>ss_id</th>
<th>vistrails_id</th>
<th>wf_version</th>
<th>ts_start</th>
<th>ts_end</th>
</tr>
</thead>
<tbody>
<tr>
<td>182</td>
<td>304</td>
<td>16</td>
<td>34</td>
<td>2006-09-11 12:59:09</td>
<td>2006-09-11 00:00:00</td>
</tr>
<tr>
<td>183</td>
<td>305</td>
<td>16</td>
<td>34</td>
<td>2006-09-11 13:02:45</td>
<td>2006-09-11 13:03:08</td>
</tr>
<tr>
<td>184</td>
<td>305</td>
<td>16</td>
<td>34</td>
<td>2006-09-11 13:03:08</td>
<td>2006-09-11 13:03:22</td>
</tr>
<tr>
<td>186</td>
<td>305</td>
<td>16</td>
<td>34</td>
<td>2006-09-11 13:03:35</td>
<td>2006-09-11 13:03:45</td>
</tr>
<tr>
<td>191</td>
<td>384</td>
<td>17</td>
<td>212</td>
<td>2006-09-11 21:00:30</td>
<td>2006-09-11 21:01:07</td>
</tr>
<tr>
<td>192</td>
<td>403</td>
<td>18</td>
<td>212</td>
<td>2006-09-12 11:31:04</td>
<td>2006-09-12 11:31:28</td>
</tr>
</tbody>
</table>
Querying and Understanding Provenance

- Sample query from Provenance Challenge:
  - Find all invocations of procedure align_warp using a twelfth order nonlinear 1365 parameter model (see model menu describing possible values of parameter "-m 12" of align_warp) that ran on a Monday.

- New provenance query language
  - workflow{*:}
    - x where x.module = AlignWarp and
    - x.parameter('model') = '12' and
    - (log{x}: y where y.dayOfWeek = 'Monday')

For details see [http://twiki.gridprovenance.org/bin/view/Challenge/VisTrails](http://twiki.gridprovenance.org/bin/view/Challenge/VisTrails)

- But who is going to write those queries?
- WYSIWYQ -- What You See Is What You Query
  - Interface to create workflow is same as to query!
Demo: Querying Provenance
Computing Workflow Differences

- No need to compute graph isomorphism!
- Vistrail is a rooted tree: all nodes have a common ancestor—differences are well-defined and *simple to compute*

\[ vt_1 = x_i \circ x_{i-1} \circ \cdots \circ x_1 \circ \emptyset \]
\[ vt_2 = x_j \circ x_{j-1} \circ \cdots \circ x_1 \circ \emptyset \]
\[ vt_1 - vt_2 = \{x_i, x_{i-1}, \ldots, x_1, \emptyset\} - \{x_j, x_{j-1}, \ldots, x_1, \emptyset\} \]

- Different semantics:
  - Exact, based on ids
  - Approximate, based on module/connection signatures
Collaborative Exploration

- Collaboration is key to data exploration
  - Translational, integrative approaches to science
- Store provenance information in a database
- Synchronize concurrent updates through locking
- Asynchronous access: similar to version control systems
  - Check out, work offline, synchronize
  - Users exchange patches
- No need for a central repository—support for distributed collaboration
  - For details see Callahan et al, SCI Institute Technical Report, No. UUSCI-2006-016 2006
Extensibility: Adding New Modules

class PythonCalc(Module):

    def compute(self):
        v1 = self.getInputFromPort("value1")
        v2 = self.getInputFromPort("value2")
        self.setResult("value", self.op(v1, v2))

    def op(self, v1, v2):
        op = self.getInputFromPort("op")
        if op == '+':
            return v1 + v2
        elif op == '-':
            return v1 - v2
        elif op == '*':
            return v1 * v2
        elif op == '/':
            return v1 / v2
        raise ModuleError('unrecognized operation: \"%s\" \% op)

    def initialize(*args, **keywords):
        reg = modules.module_registry
        reg.addModule(PythonCalc)
        reg.addInputPort(PythonCalc, "value1", (modules.basic_modules.Float, 'the first argument'))
        reg.addInputPort(PythonCalc, "value2", (modules.basic_modules.Float, 'the second argument'))
        reg.addInputPort(PythonCalc, "op", (modules.basic_modules.String, 'the operation'))
        reg.addOutputPort(PythonCalc, "value", (modules.basic_modules.Float, 'the result'))

Define module

Register with VisTrails

In your .vistrails:
addPackage('pythonCalc')
Conclusions and Future Work

- Provenance beyond reproducibility: support and streamline scientific process
  - Reduce time to insight!
- Initial focus on visualization, but ideas are applicable to exploratory tasks in general
  - Easy to extend (all python, support web services too!)
- Many important applications in different domains—some ongoing collaborations:
  - OHSU (environmental observation and forecasting systems); Emulab (Networking experiments); Harvard Medical School (radiation oncology); UCSD (biomedical informatics)
- Automate the generation of data products, e.g., by analogy
Automating Workflow Creation: Visualization by Analogy

By analogy, specialist can do it!
Automating Workflow Creation: Visualization by Analogy

By analogy, specialist can do it!

Simple in VisTrails:

\[ v_4 = (v_2 - v_1) \circ v_3 \]
Conclusions and Future Work

- Provenance beyond reproducibility: support and streamline scientific process
  - Reduce time to insight!
- Initial focus on visualization, but ideas are applicable to exploratory tasks in general
- Many important applications in different domains—some ongoing collaborations:
  - OHSU (environmental observation and forecasting systems); Emulab (Networking experiments); Harvard Medical School (radiation oncology); UCSD (biomedical informatics)
-Automate the generation of data products, e.g., by analogy
- Support additional workflow execution engines
  - Collaborating with Kepler
- Mine history—potentially useful information about good and bad problem-solving strategies
- Vision: scientists (end-users) steering their own explorations
Acknowledgements

- This work is partially supported by the National Science Foundation, the Department of Energy, an IBM Faculty Award, and a University of Utah Seed Grant.
More info about VisTrails

google vistrails

Or

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