Managing Exploratory Workflows

Juliana Freire

Cláudio T. Silva

http://vistrails.sci.utah.edu

University of Utah

Joint work with: Erik Andersen, Steven P. Callahan, David Koop, Emanuele Santos, Carlos E. Scheidegger, Nathan Smith, 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
- Workflows are rapidly replacing primitive *shell* scripts
  - *Kepler*, *Taverna*, Apple's Mac OS X Automator, Microsoft Windows Workflow Foundation, and SGI Scientific Workflow Solution,
- 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
  - E.g., Data mining, visualization, visual analytics, simulation, etc.

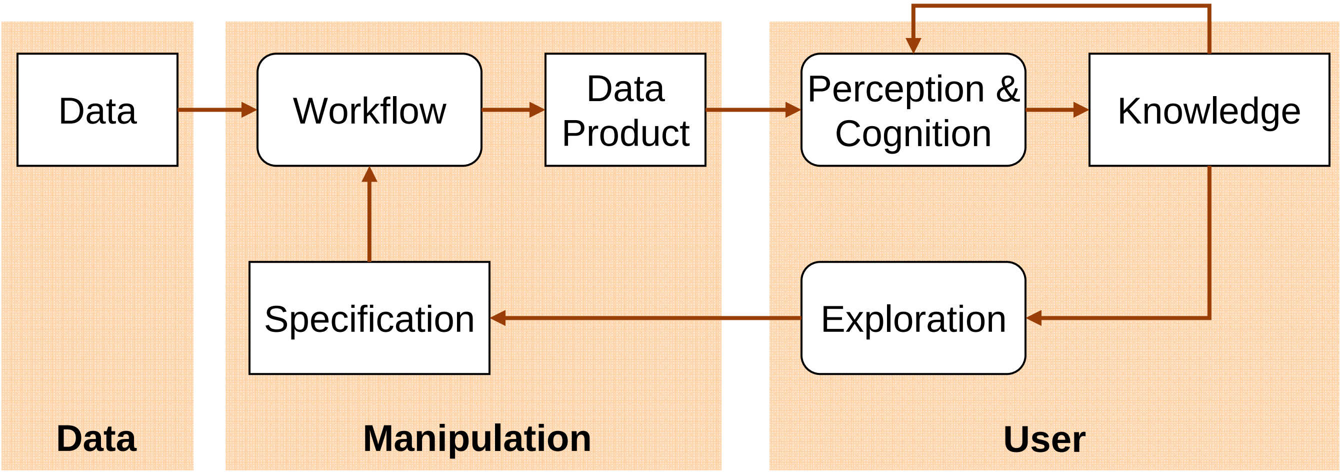
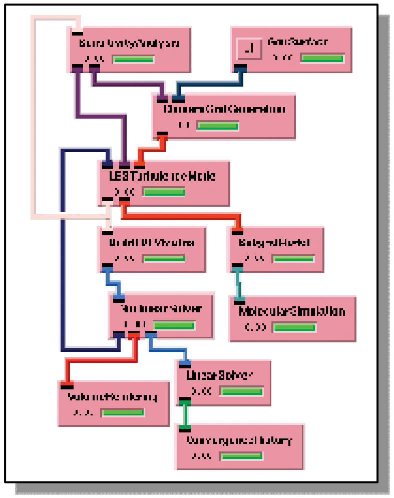


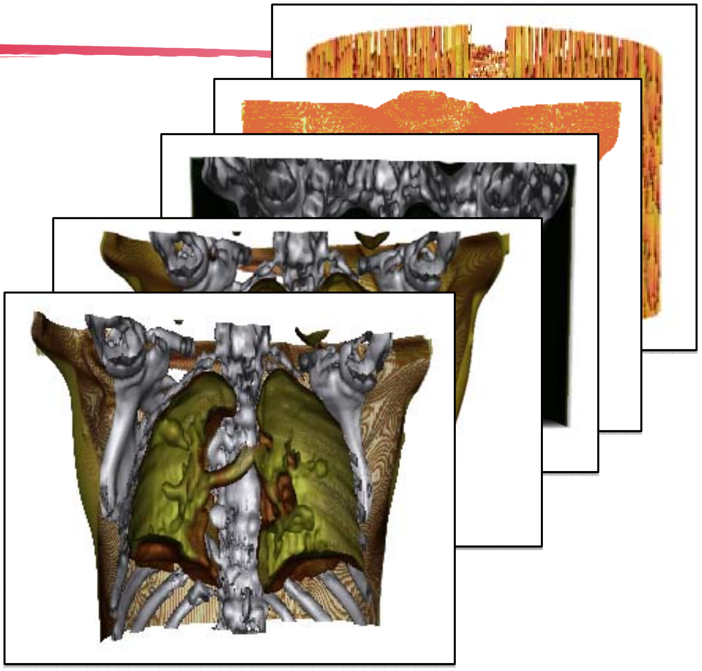
Figure modified from J. van Wijk, IEEE Vis 2005

Data Exploration and Workflows: Today

raw data

dataflow





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
v
wi

Added
te

Added plane
to
i
s

Found good
t
f

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 and process management!

VisTrails: Managing Exploration

- Streamlines the creation, execution and sharing of a large number complex workflows
- 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

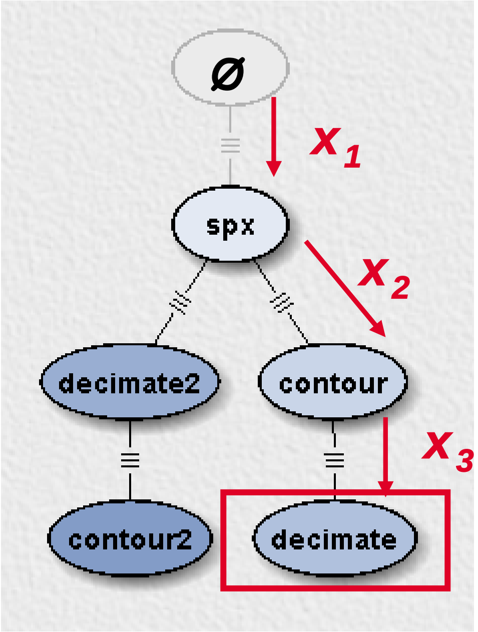
- Change-based provenance
- Scalable generation of data products
- Interacting with and querying provenance
- Extensibility

- More details available in

  http://vistrails.sci.utah.edu

Change-Based Provenance

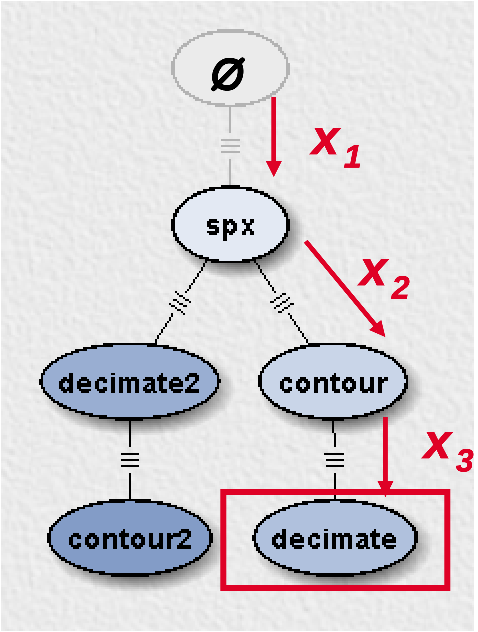
- Records user interactions with workflows
- Workflow evolution is captured in a *vistrail*—a rooted tree where
  - *nodes* correspond to workflow versions
  - *edges* correspond to actions that transform the parent into the child workflow



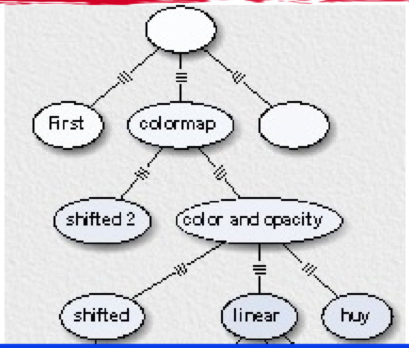
$$decimate = x_3 \circ x_2 \circ x_1 \circ \varnothing$$

Change-Based Provenance

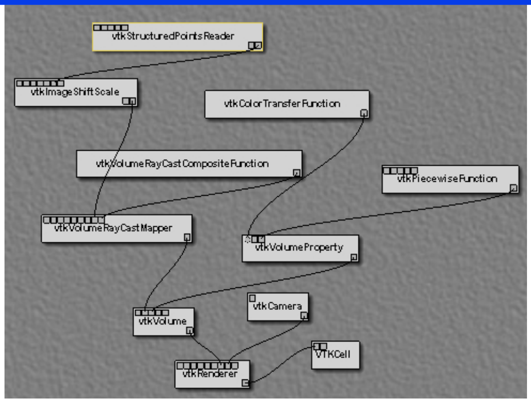
- Records user interactions with workflows
- Workflow evolution is captured in a *vistrail*—a rooted tree where
  - *nodes* correspond to workflow versions
  - *edges* correspond to actions that transform the parent into the child workflow
- Action algebra:
  - addModule, deleteModule, addConnection,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



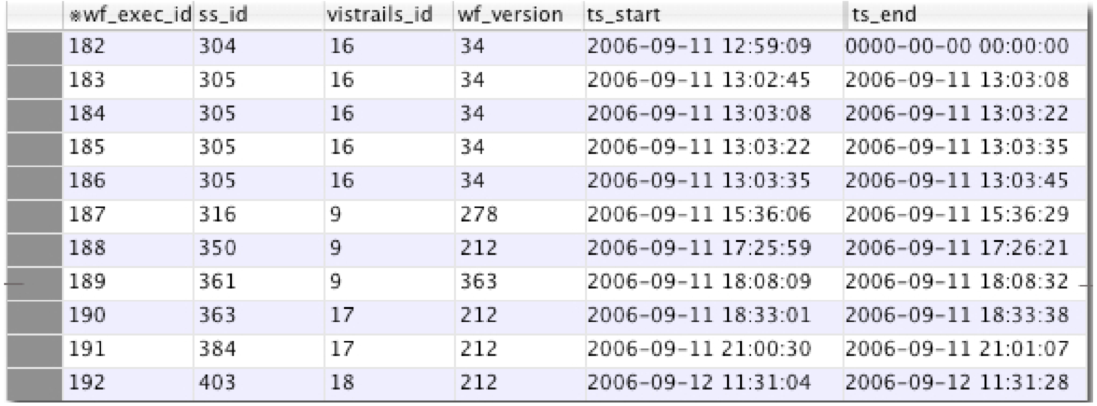
Three Layers of Metadata



Workflow Evolution



Workflow



*wf_exec_id	ss_id	vistrails_id	wf_version	ts_start	ts_end
182	304	16	34	2006-09-11 12:59:09	0000-00-00 00:00:00
183	305	16	34	2006-09-11 13:02:45	2006-09-11 13:03:08
184	305	16	34	2006-09-11 13:03:08	2006-09-11 13:03:22
185	305	16	34	2006-09-11 13:03:22	2006-09-11 13:03:35
186	305	16	34	2006-09-11 13:03:35	2006-09-11 13:03:45
187	316	9	278	2006-09-11 15:36:06	2006-09-11 15:36:29
188	350	9	212	2006-09-11 17:25:59	2006-09-11 17:26:21
189	361	9	363	2006-09-11 18:08:09	2006-09-11 18:08:32
190	363	17	212	2006-09-11 18:33:01	2006-09-11 18:33:38
191	384	17	212	2006-09-11 21:00:30	2006-09-11 21:01:07
192	403	18	212	2006-09-12 11:31:04	2006-09-12 11:31:28

Execution

Querying and Understanding Provenance

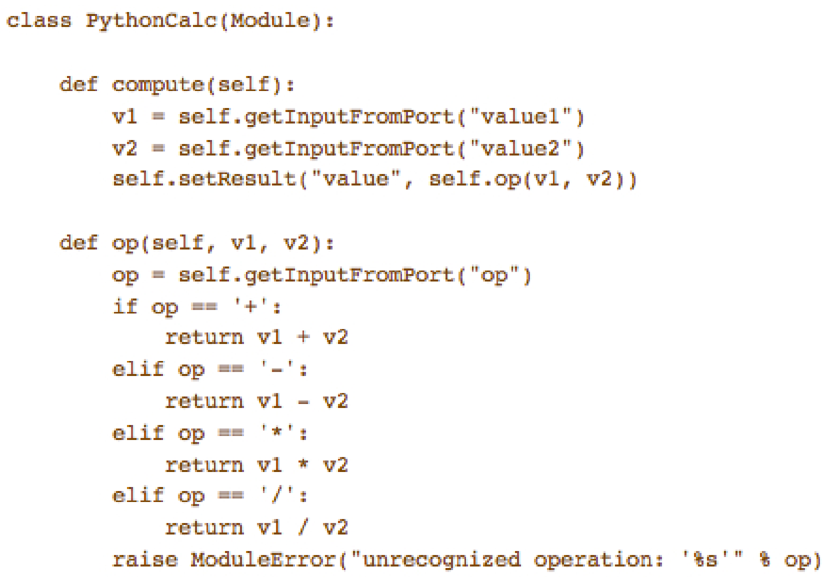
- Sample query from *Provenance Challenge*:
  - Find all invocations of procedure align_warp using a twelfth order nonlinear 1365 parameter model (see model menu describing possible values of parameter "-m 12" of align_warp) that ran on a Monday.
- New provenance query language

```
– wf{*}:                                            →  Workflow Evolution
x where x.module = AlignWarp and
x.parameter('model') = '12' and                     →  Workflow
(log{x}: y where y.dayOfWeek = 'Monday')            →  Execution
```

For details see 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!

Extensibility: Adding New Modules

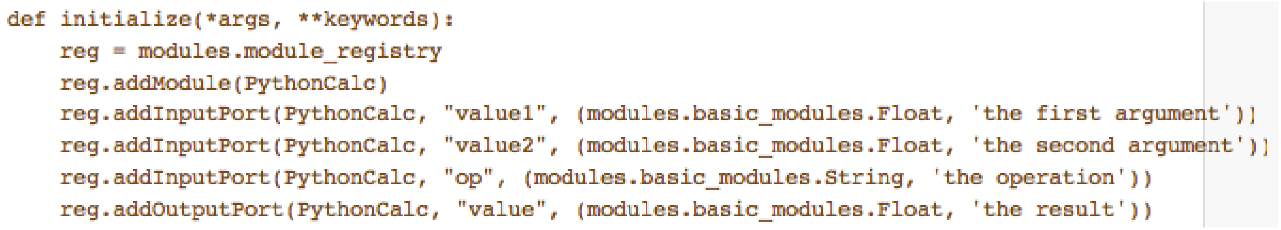


```
class PythonCalc(Module):

    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)
```

Define module

Register with VisTrails

In your .vistrails:
addPackage('pythonCalc')

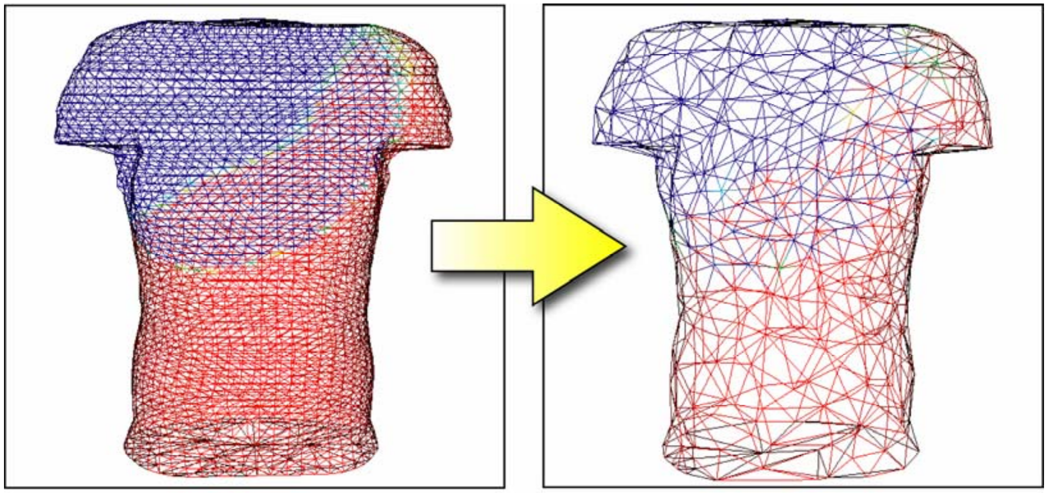


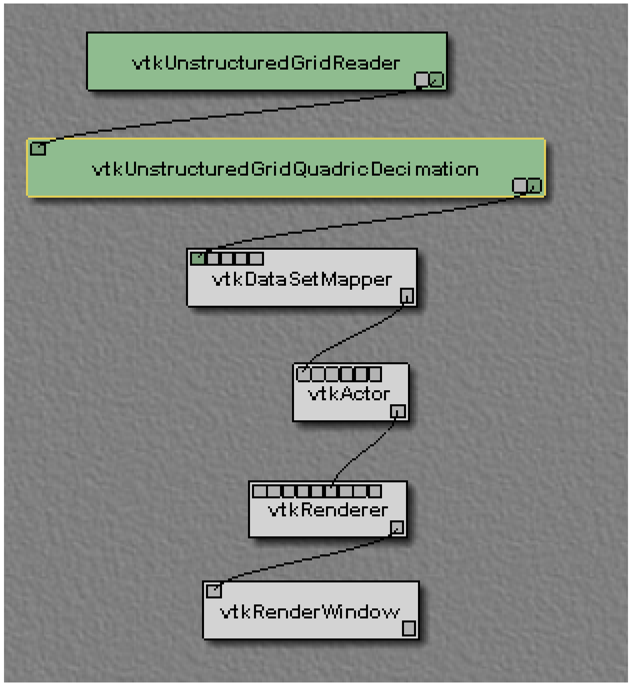
```
def initialize(*args, **keywords):
    reg = modules.module_registry
    reg.addModule(PythonCalc)
    reg.addInputPort(PythonCalc, "value1", (modules.basic_modules.Float, 'the first argument'))
    reg.addInputPort(PythonCalc, "value2", (modules.basic_modules.Float, 'the second argument'))
    reg.addInputPort(PythonCalc,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

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

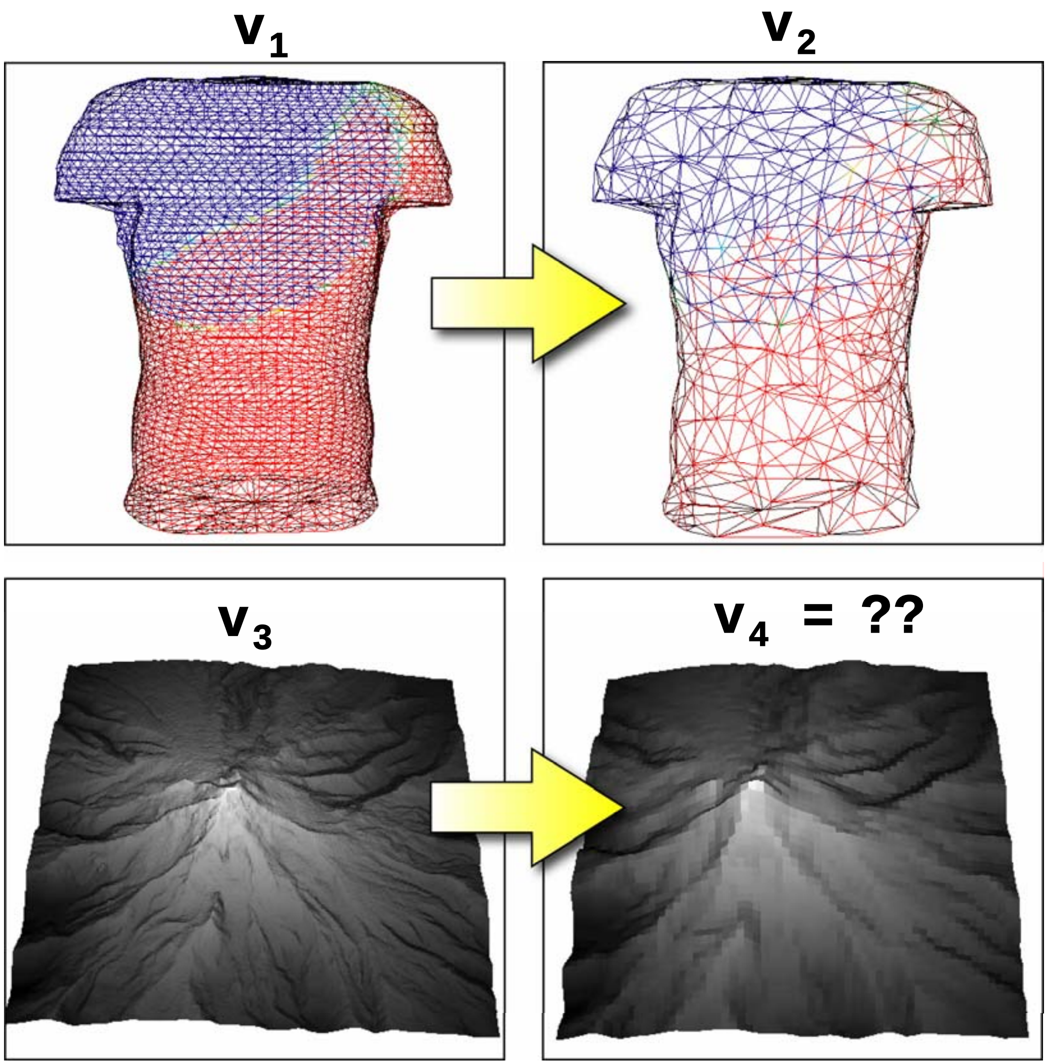
Automating Workflow Creation: Visualization by Analogy

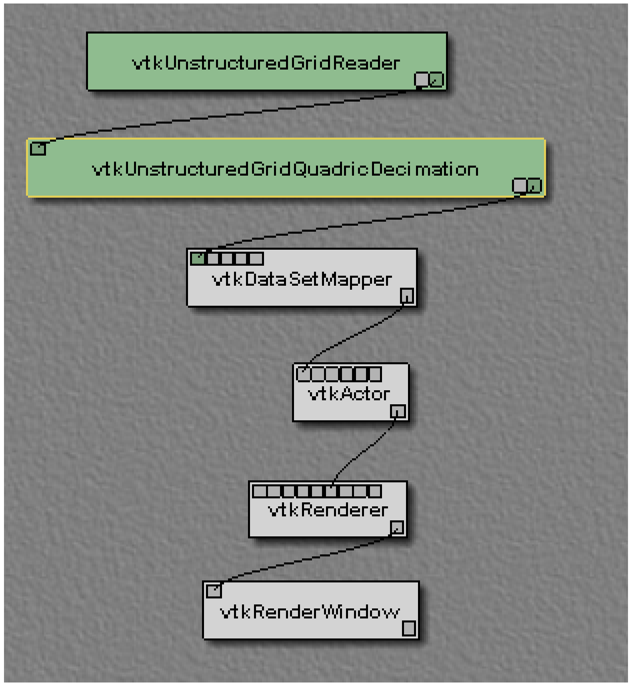



vtkUnstructuredGridReader
vtkUnstructuredGridQuadricDecimation
vtkDataSetMapper
vtkActor
vtkRenderer
vtkRenderWindow

By analogy, specialist can do it!

Automating Workflow Creation: Visualization by Analogy




vtkUnstructuredGridReader
vtkUnstructuredGridQuadricDecimation
vtkDataSetMapper
vtkActor
vtkRenderer
vtkRenderWindow

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; execute workflows on the grid
- Scalable database backend
- 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://vistrails.sci.utah.edu