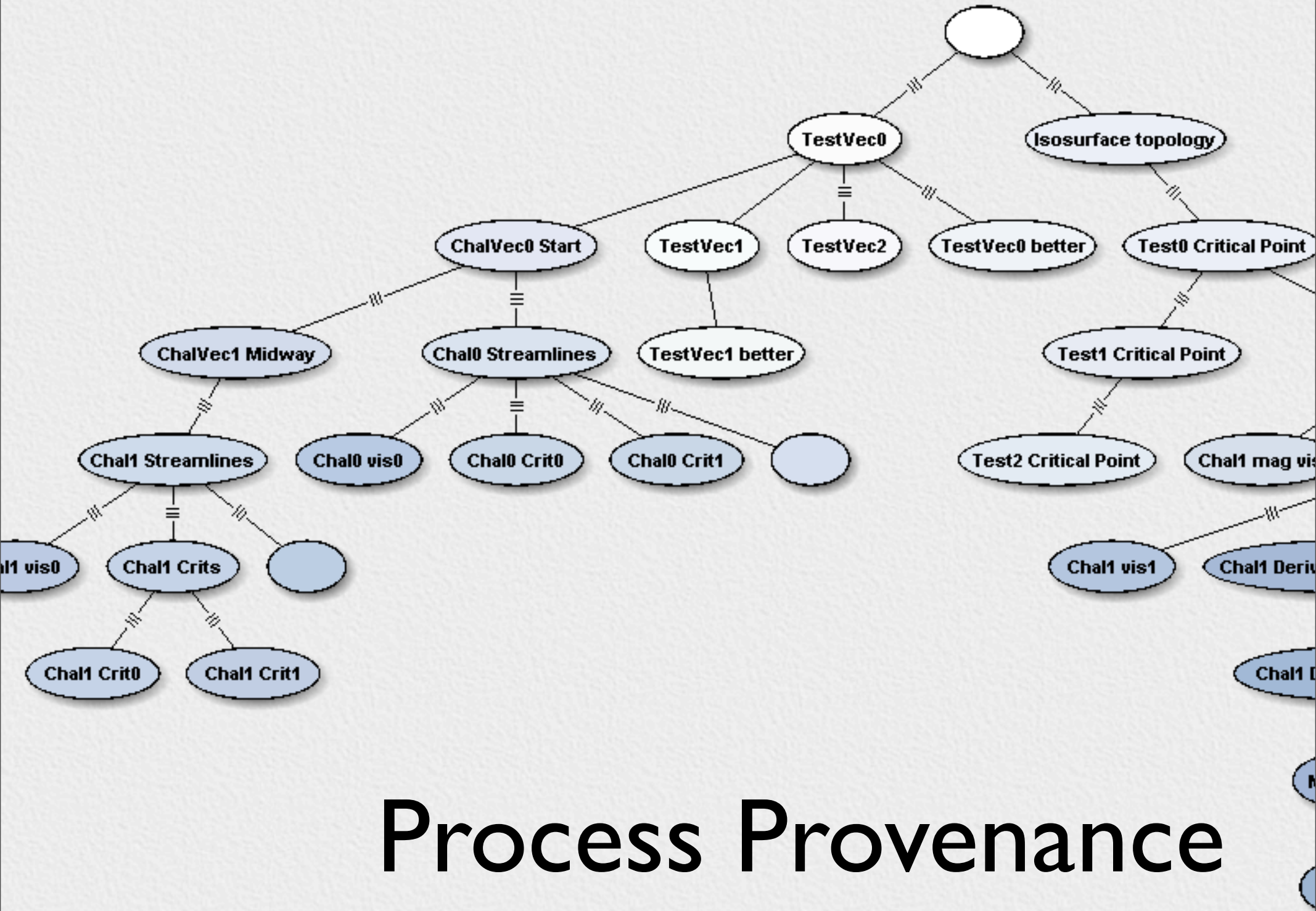
VisTrails: Process Provenance for SciVis

GGF18 Provenance Challenge Entry

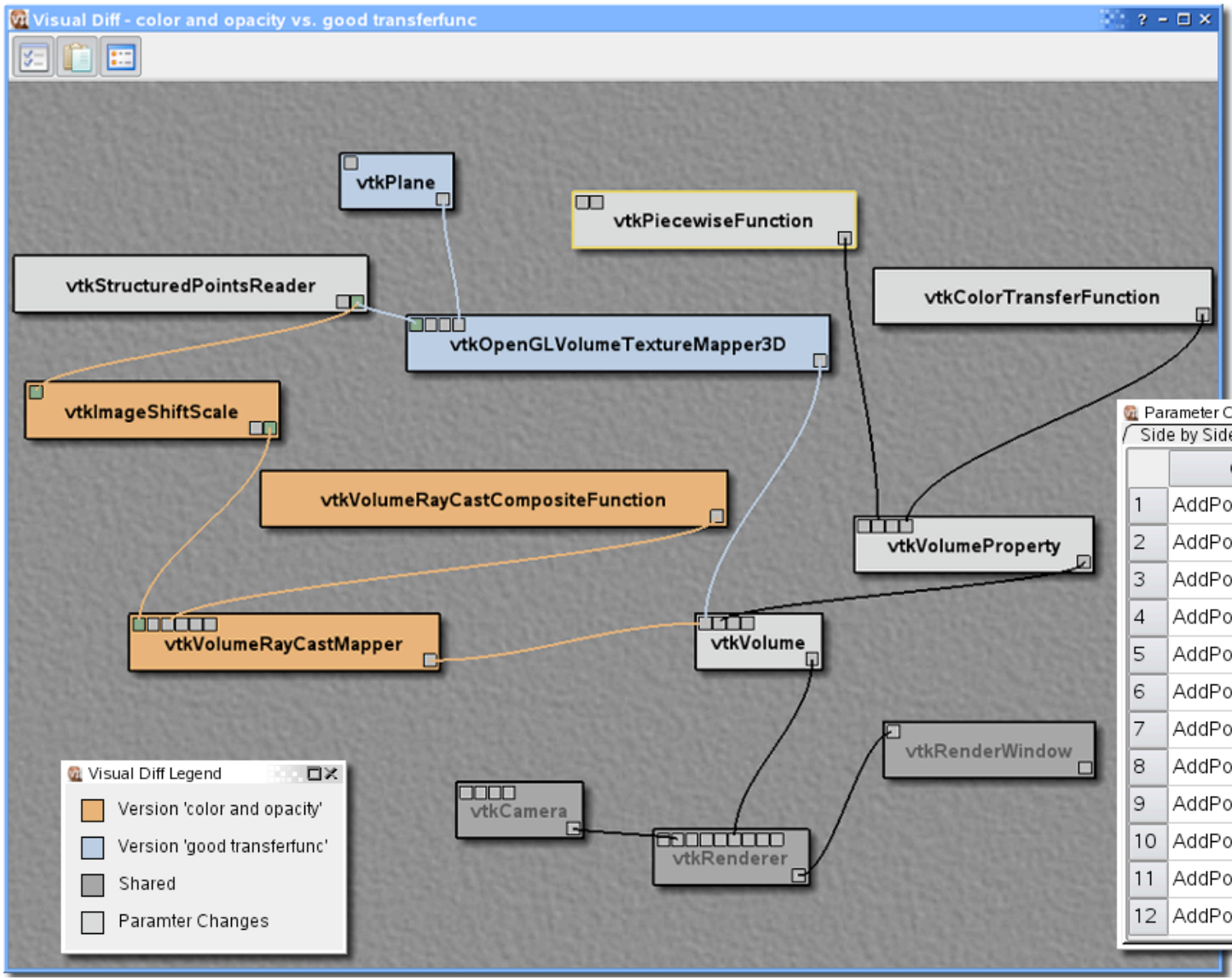
Scientific Computing and Imaging Institute, School of Computing - University of Utah

Erik Anderson, Steve Callahan, Juliana Freire, Emanuele Santos, Cláudio Silva, Carlos Scheidegger, Huy Vo



Process Provenance

Process Evolution

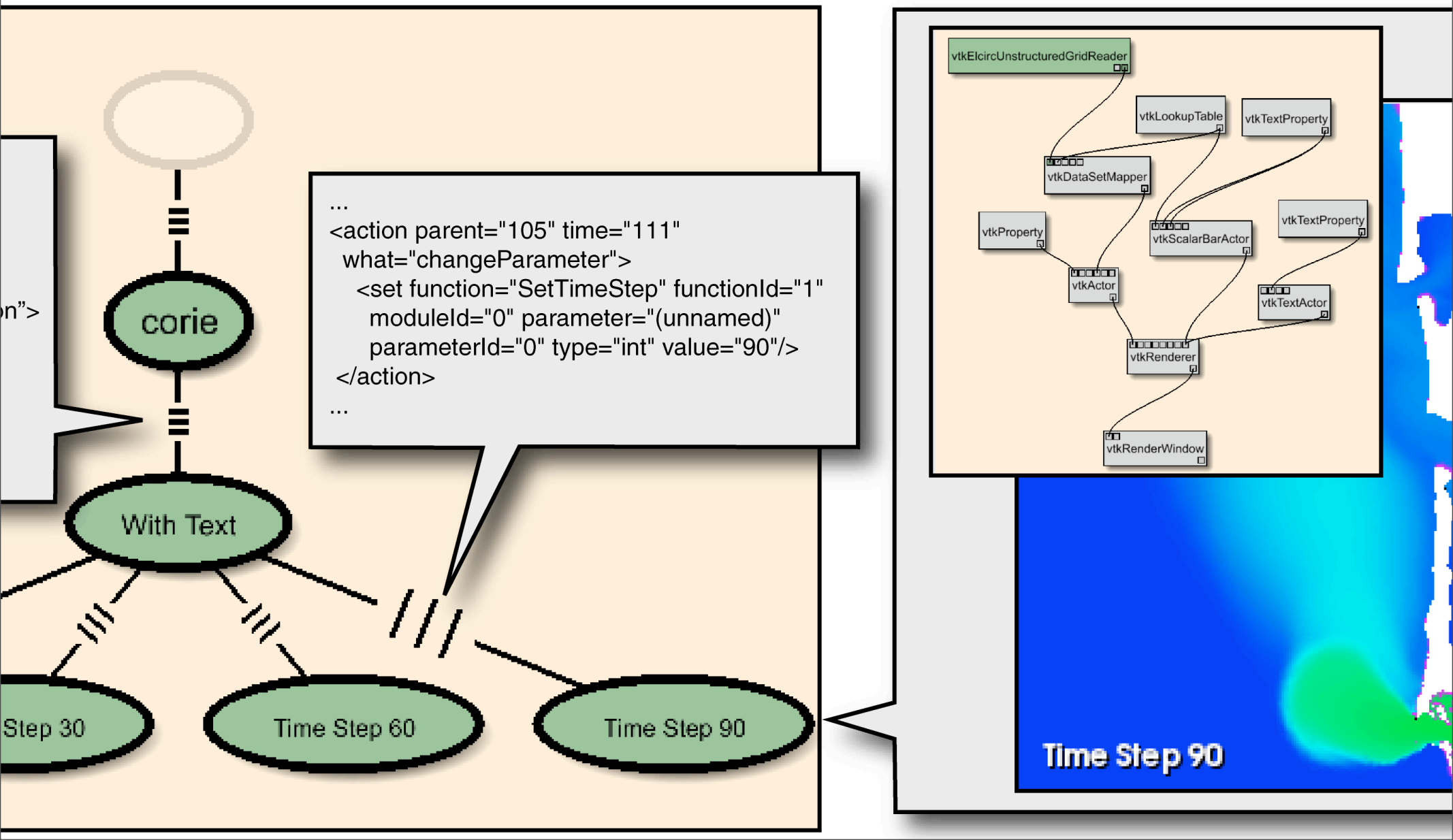


Parameter Changes - vtkPiecewiseFunction

Side by Side | Filtered | Full

	color and opacity	good transferfunc
1	AddPoint(0.0,0.0)	AddPoint(-1431.0,0.934)
2	AddPoint(104.0,0.515)	AddPoint(-1439.0,0.0)
3	AddPoint(109.0,0.964)	AddPoint(-161.0,0.855)
4	AddPoint(117.0,0.855)	AddPoint(-293.0,0.965)
5	AddPoint(127.0,0.0)	AddPoint(-379.0,0.516)
6	AddPoint(166.0,0.0)	AddPoint(-5.0,0.0)
7	AddPoint(167.0,1.0)	AddPoint(-745.0,0.516)
8	AddPoint(255.0,1.0)	AddPoint(-854.0,0.934)
9	AddPoint(36.0,0.0)	AddPoint(0.0,0.0)
10	AddPoint(37.0,0.934)	AddPoint(1990.0,1.0)
11	AddPoint(74.0,0.934)	AddPoint(610.0,0.0)
12	AddPoint(80.0,0.515)	AddPoint(610.0,1.0)

Version Tree, Workflows

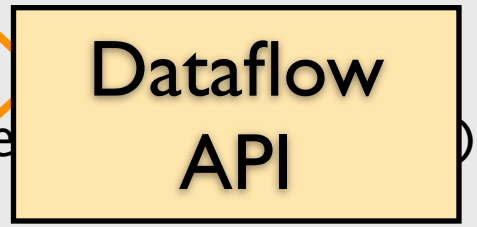


The VisTrails Plugin architecture

```
class AlignWarp(ProvenanceChallenge):

    def compute(self):
        image = self.getInputFromPort("image")
        ref   = self.getInputFromPort("reference")
        model = self.getInputFromPort("model")
        o     = self.interpreter.filePool.createFile
        cmd   = self.air_cmd_line('align_warp',
                                  image.name, ref.name, o.name,
                                  '-m', model, '-q')

        self.run(cmd)
        self.setResult("output", o)
```



Each module is just Python source

The VisTrails Plugin architecture

```
class AIRHeaderFile(modules.basic_modules.File):

    def get_header_annotations(self):
        ....

    def compute(self):
        modules.basic_modules.File.compute(self)
        d = self.get_header_annotations()
        self.annotate(d)
```

Annotation API

Each module is just Python source

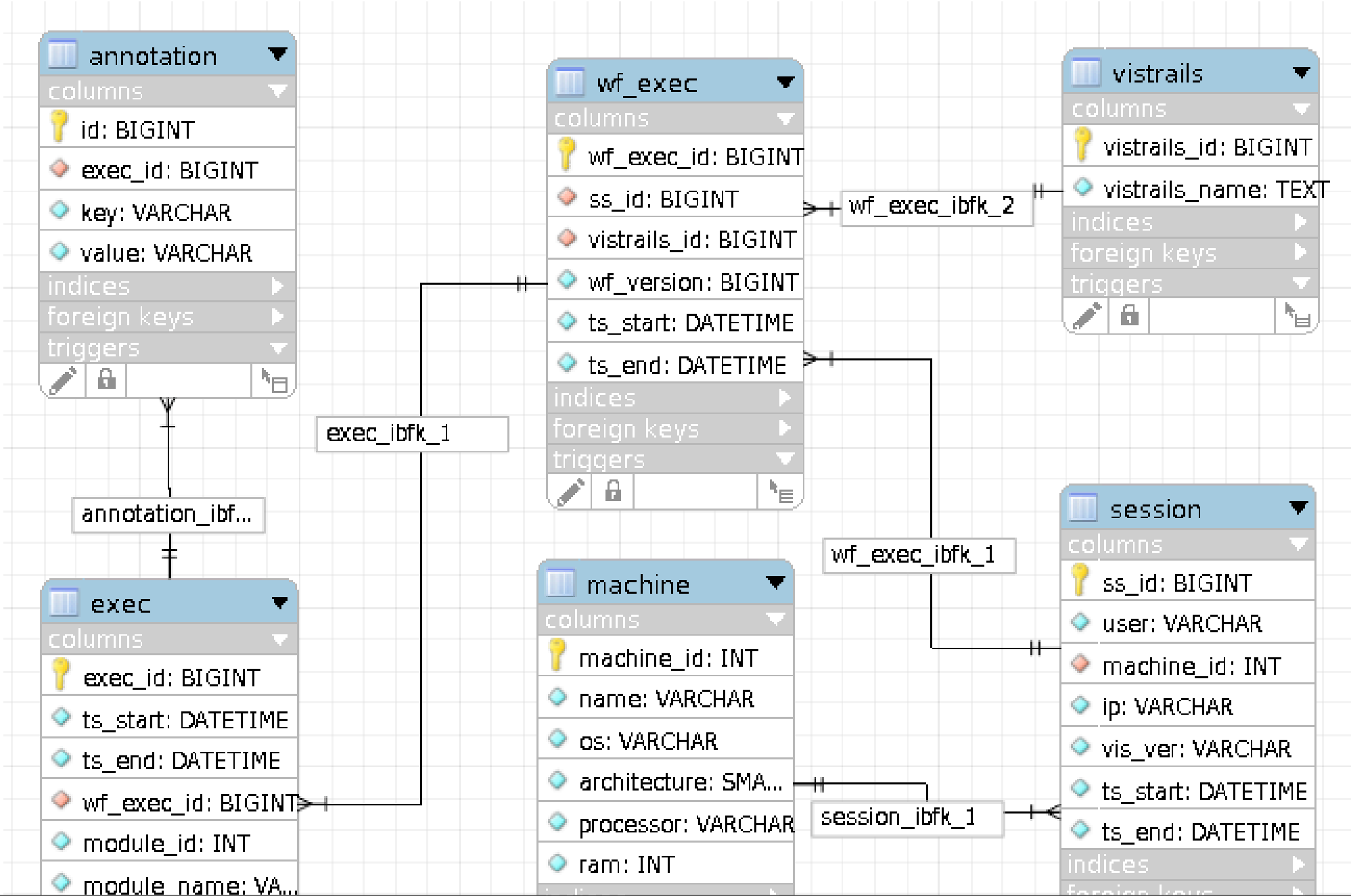
The VisTrails Plugin architecture

```
addModule(AlignWarp)
addInputPort(AlignWarp, "image", ...)
addInputPort(AlignWarp, "image_header", ...)
addInputPort(AlignWarp, "reference", ...)
addInputPort(AlignWarp, "reference_header", ...)
addInputPort(AlignWarp, "model", ...)
addOutputPort(AlignWarp, "output", ...)
```

Registering a module within VisTrails

Each module is just Python source

DB for runtime info

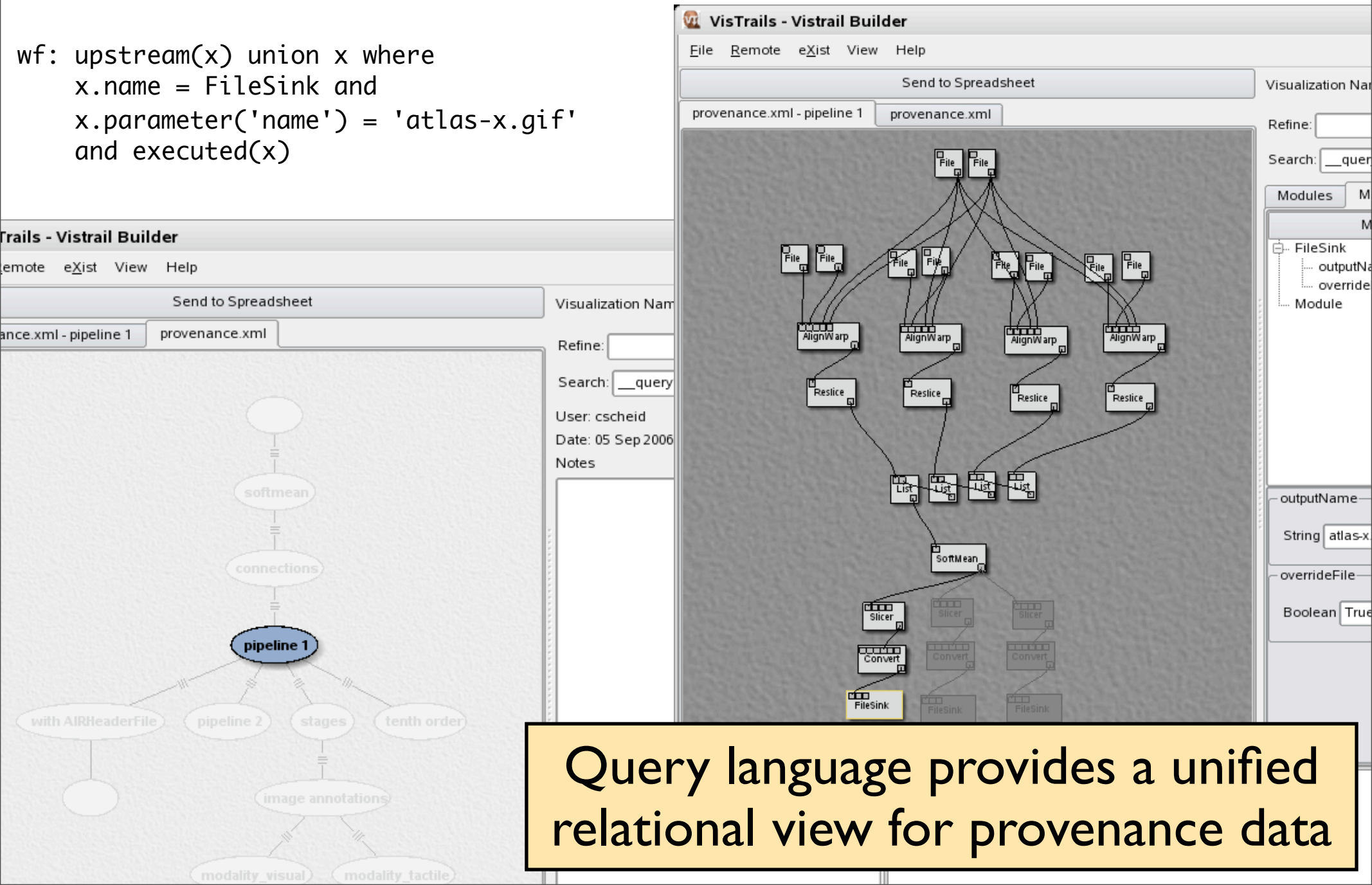


Provenance Trace

- Version tree induces a relation (version, modules)
- We extended the provenance with new relations that capture executions of the workflows and modules
- How do we query these?

Provenance Trace

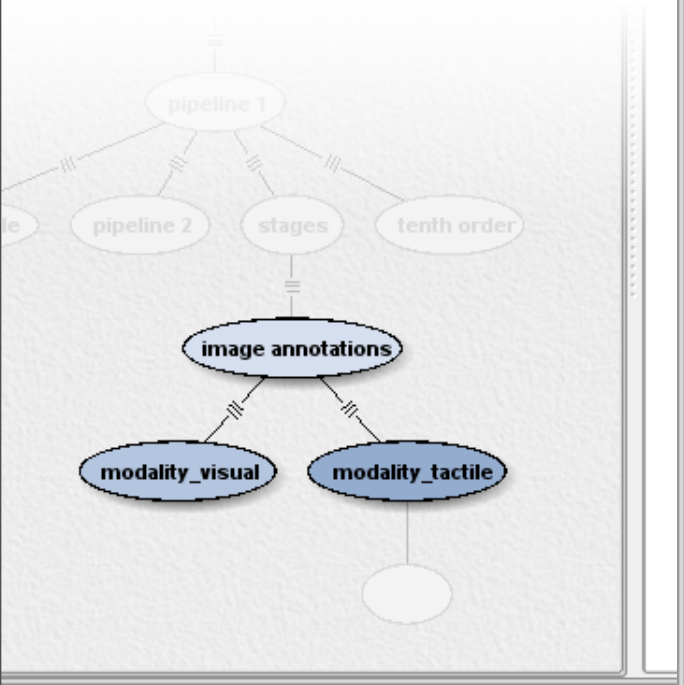
```
wf: upstream(x) union x where
    x.name = FileSink and
    x.parameter('name') = 'atlas-x.gif'
    and executed(x)
```

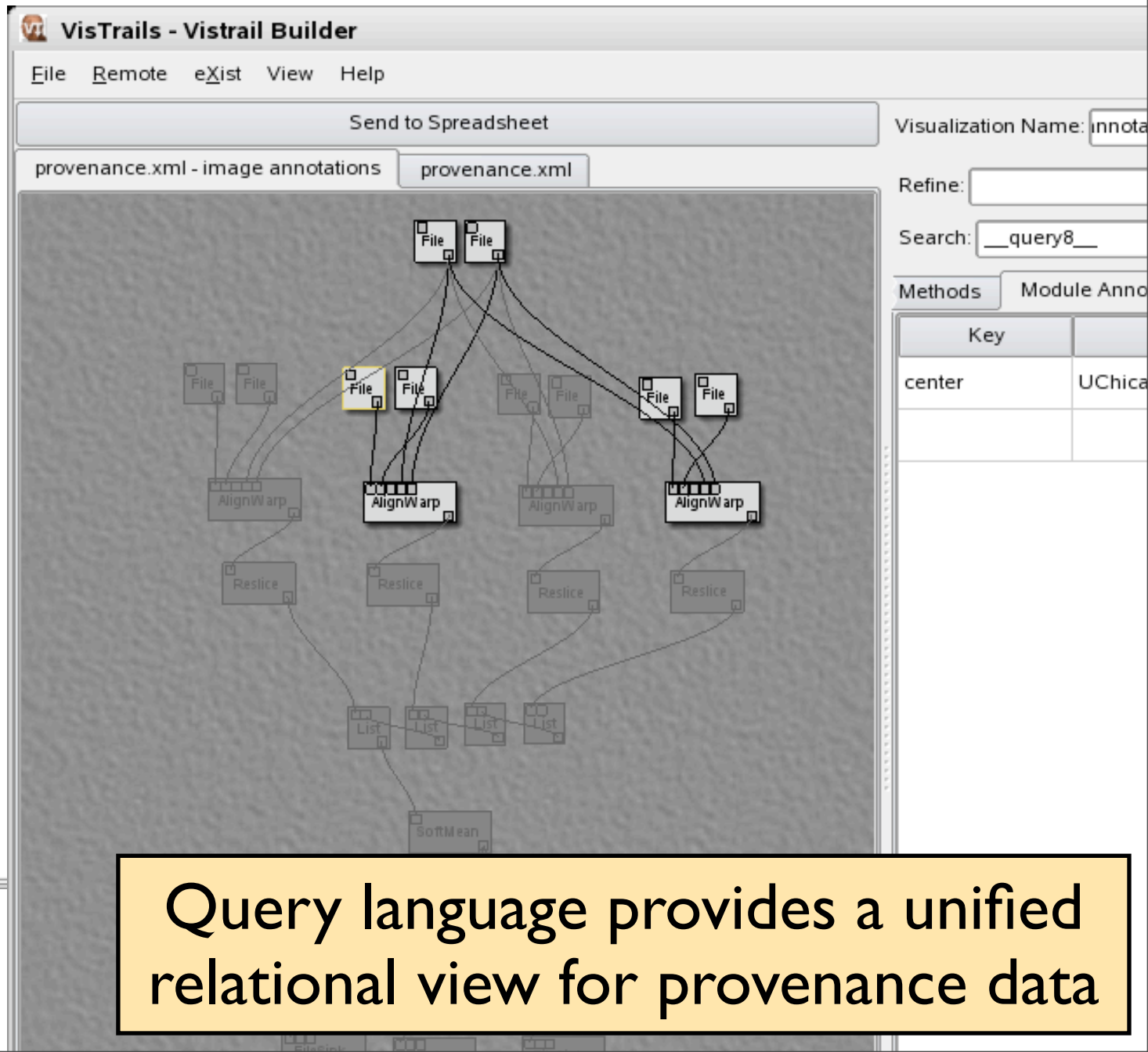


Query language provides a unified relational view for provenance data

Provenance Trace

```
wf: upstream(x) union x
where x.module = AlignWarp
and y in inputs(x)
and y.annotation('center') =
        'UChicago'
```





Query language provides a unified relational view for provenance data

The data logging problem

- Users really want not only execution statisticts, but also the (intermediate) results
- Sufficiently general modules require (at least) kernel-level control of file creation
  - Undecidable otherwise!
- Our solution: provide an API modules can use at runtime to store metadata for further querying

Acknowledgments

- Workshop organizers
- VisTrails team
- NSF (IIS-0513692, CCF-0401498, EIA-0323604, CNS-0514485, IIS-053468, CNS-0528201, OISE-0405402), DOE, IBM, CAPES/Fulbright