Cornell University
Center for Advanced Computing

Workflows and Provenance

Drew Dolgert
CAC

What You Do is a Workflow

Transcription Profiling

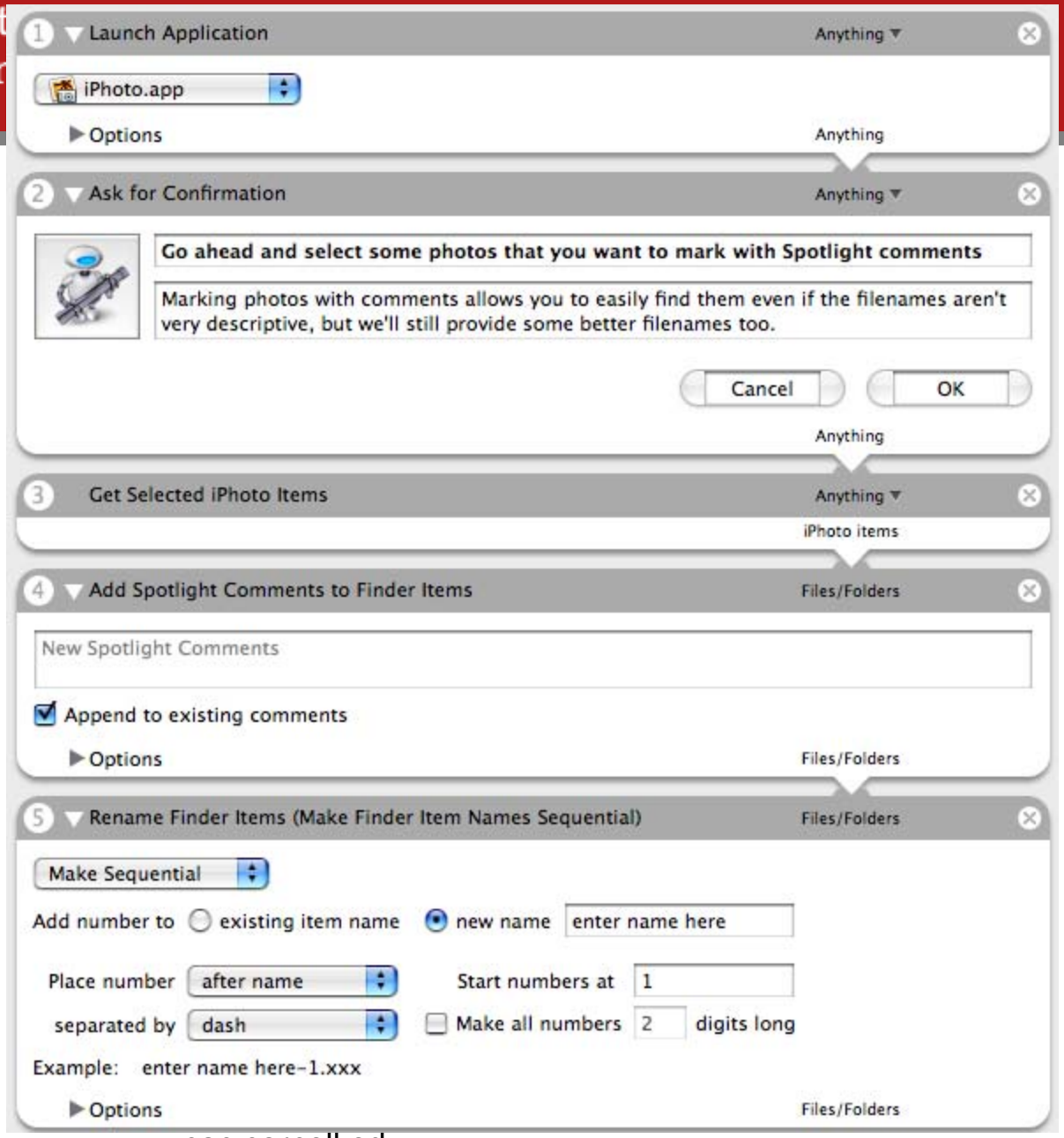
- Grow plants
- Sequence samples
- Pipeline samples
- Align to genome
- Assemble
- Integrate

- Store in
  - Executable
  - Scripts
  - Notebook

So why change?

OS X Automator

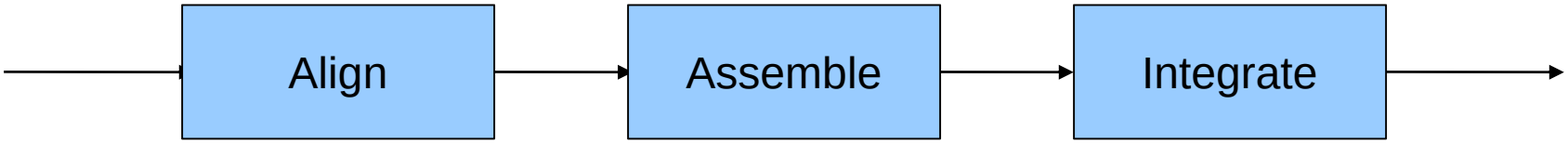
- Workflows making their way into lots of places



From http://www.macdevcenter.com/pub/a/mac/2005/09/06/what-is-automator.html?CMP=OTC-13IV03560550

A Model

Process, job, actor, transition, procedure, thorn, activity, element, unit, module



- Clear inputs and outputs
- No side-effects
- Some clear dependency (directed)
- This model can be complex
  - inside a module,
  - in how they connect,
  - in how they iterate

Vertex, edge, pipe, message, data or control dependency

How could you use this model to help computations?

Workflow Advantages

- Complex tasks from simpler ones.
- Various levels of detail.
- Systematic recording for
  - Automation
  - Reproducibility
  - Result sharing
- A program is to a workflow as a data file is to a database.

New Buttons To Push

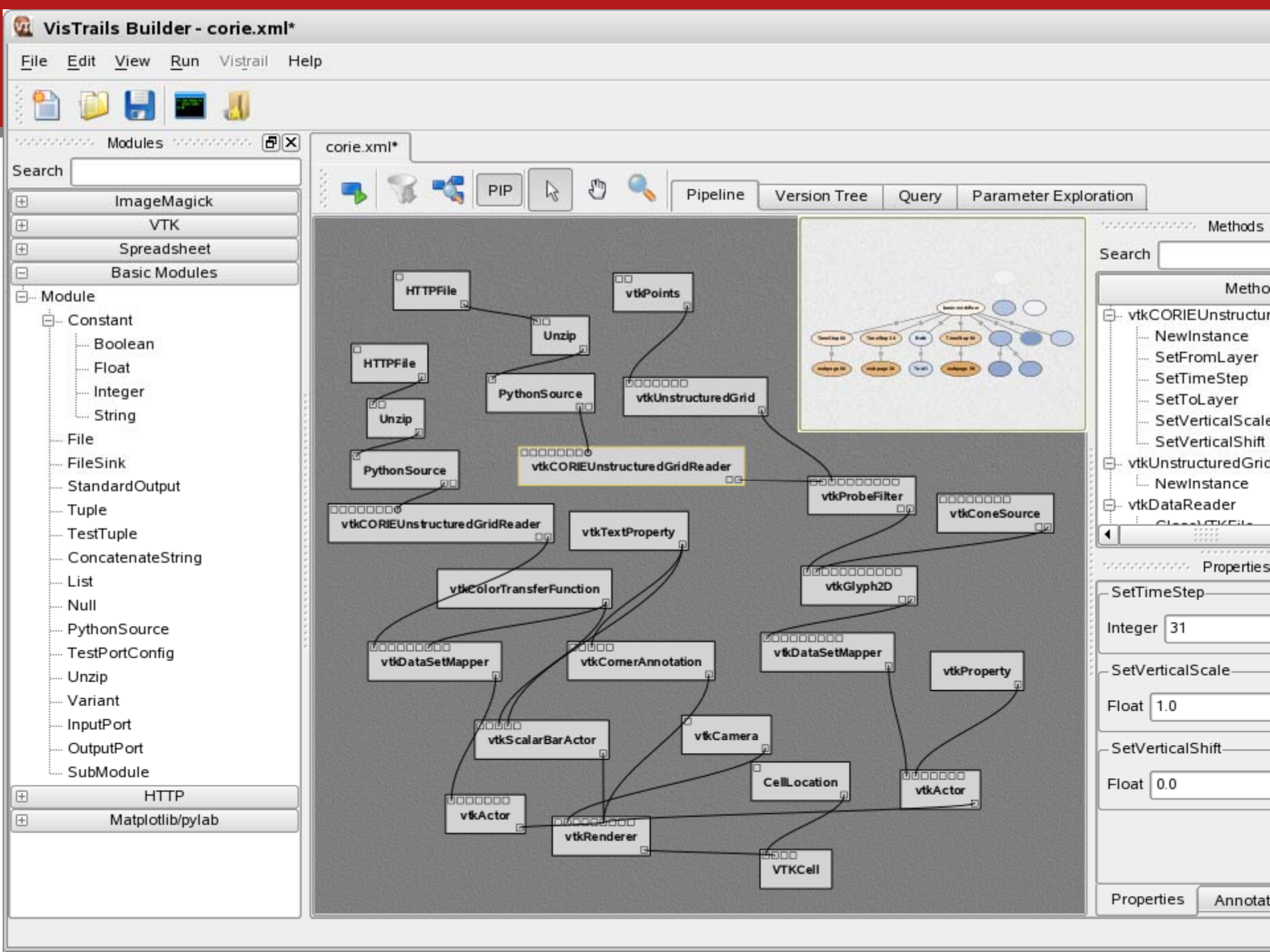
- Before/after execution can record state
- Error control
- Record of execution times, where executed
- Threading/process control done for you

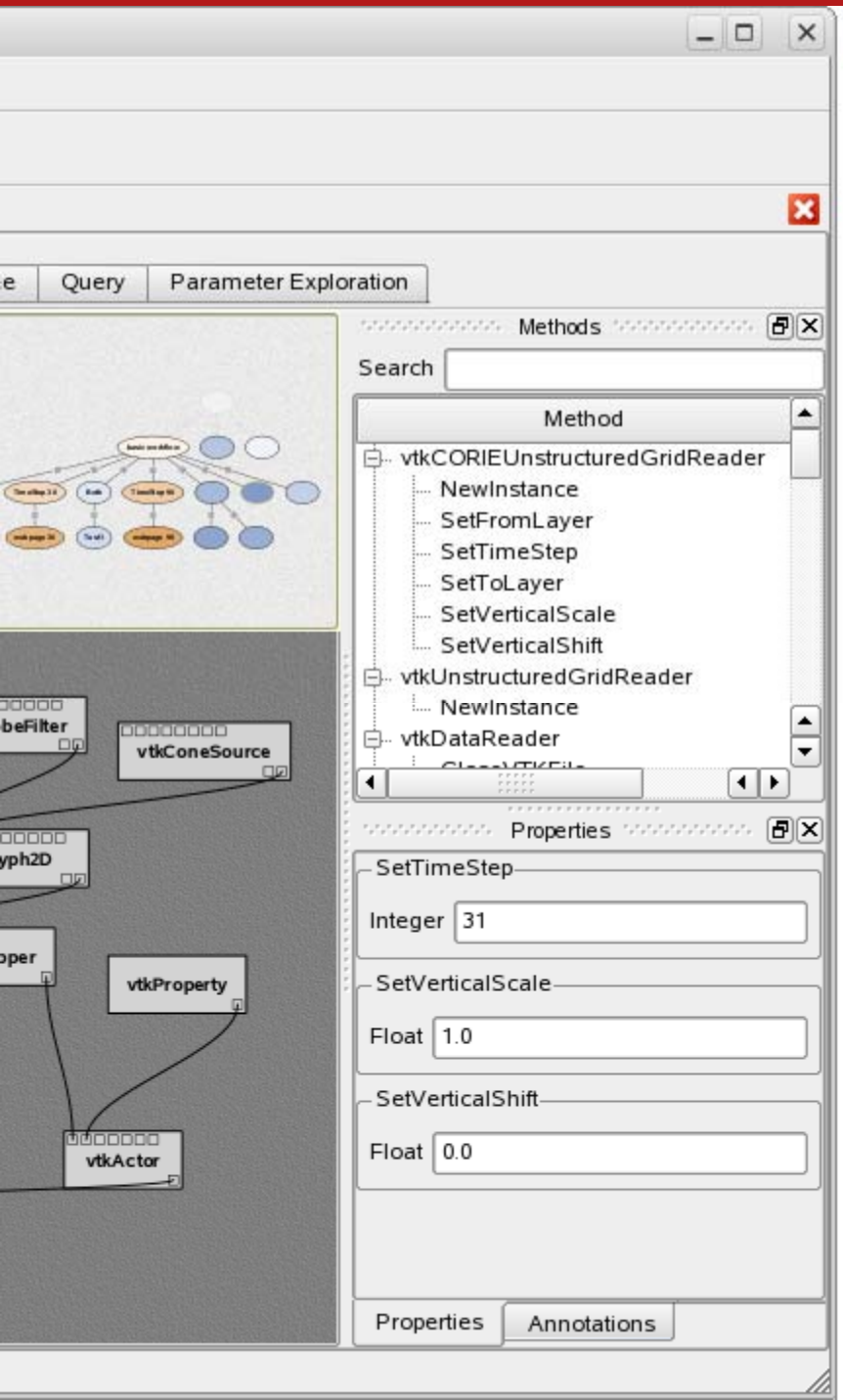
Go Do It

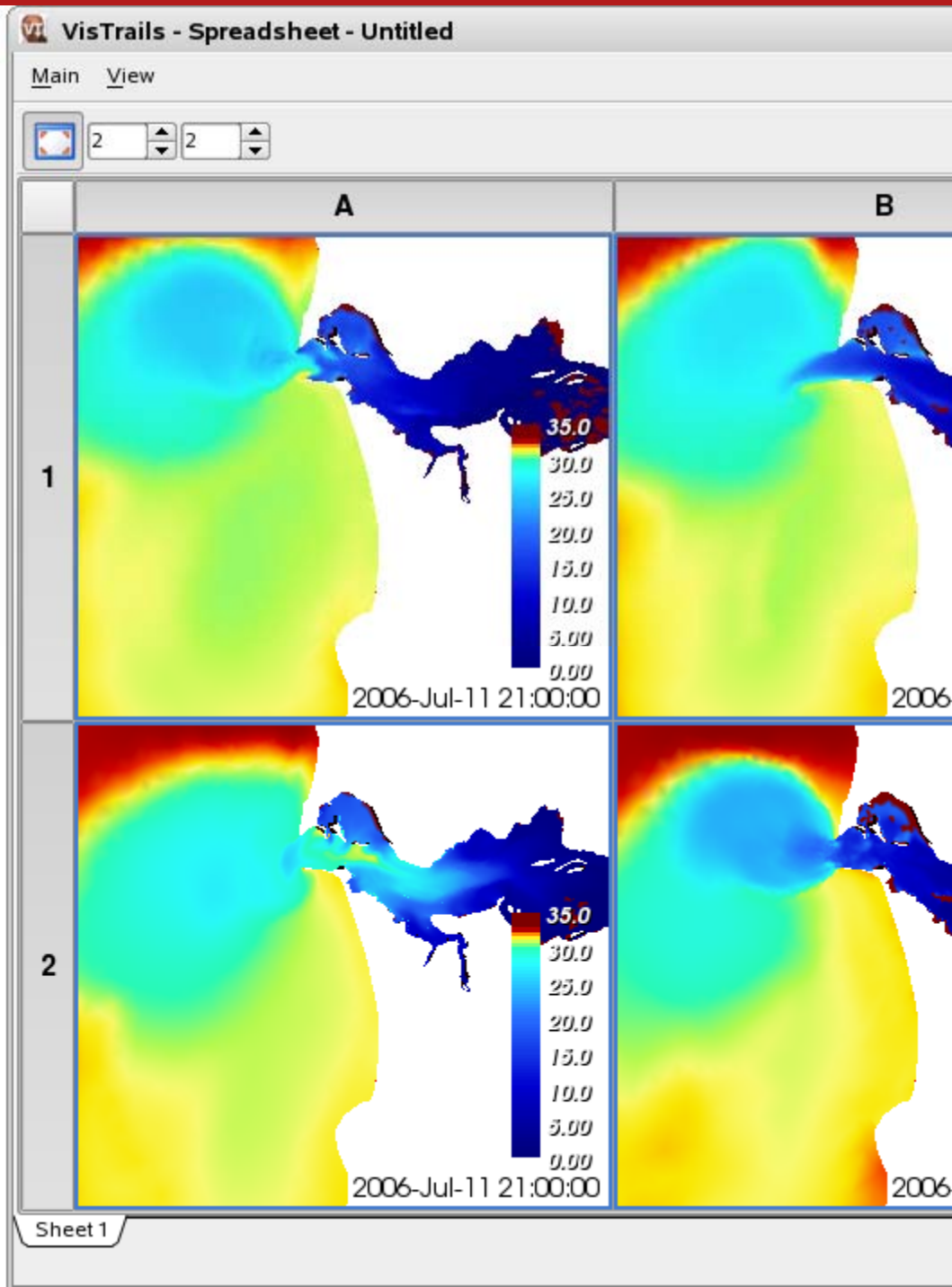
Run Again

I Did What?

- So you don't have to write the modules
- Programming without programming
- Reliability
- Ease of expression


VisTrails Builder - corie.xml*
File
Edit
View
Run
Vistrail
Help
Modules
Search
ImageMagick
VTK
Spreadsheet
Basic Modules
Module
Constant
Boolean
Float
Integer
String
File
FileSink
StandardOutput
Tuple
TestTuple
ConcatenateString
List
Null
PythonSource
TestPortConfig
Unzip
Variant
InputPort
OutputPort
SubModule
HTTP
Matplotlib/pylab
corie.xml*
PIP
Pipeline
Version Tree
Query
Parameter Exploration
HTTPFile
vtkPoints
Unzip
HTTPFile
PythonSource
vtkUnstructuredGrid
Unzip
PythonSource
vtkCORIEUnstructuredGridReader
vtkProbeFilter
vtkCORIEUnstructuredGridReader
vtkTextProperty
vtkConeSource
vtkColorTransferFunction
vtkGlyph2D
vtkDataSetMapper
vtkCornerAnnotation
vtkDataSetMapper
vtkProperty
vtkScalarBarActor
vtkCamera
CellLocation
vtkActor
vtkActor
vtkRenderer
VTKCell
Methods
Search
vtkCORIEUnstructur
NewInstance
SetFromLayer
SetTimeStep
SetToLayer
SetVerticalScale
SetVerticalShift
vtkUnstructuredGrid
NewInstance
vtkDataReader
Properties
SetTimeStep
Integer
31
SetVerticalScale
Float
1.0
SetVerticalShift
Float
0.0
Properties
Annotat


Query
Parameter Exploration
Methods
Search
Method
vtkCORIEUnstructuredGridReader
NewInstance
SetFromLayer
SetTimeStep
SetToLayer
SetVerticalScale
SetVerticalShift
vtkUnstructuredGridReader
NewInstance
vtkDataReader
vtkConeSource
vtkProperty
vtkActor
Properties
SetTimeStep
Integer 31
SetVerticalScale
Float 1.0
SetVerticalShift
Float 0.0
Properties
Annotations


VisTrails - Spreadsheet - Untitled
Main
View
A
B
1
2
35.0
30.0
25.0
20.0
15.0
10.0
5.00
0.00
2006-Jul-11 21:00:00
Sheet 1

VisTrails – The Official

- Visualization tool
- Open Source
- Allies workflow and provenance
- Python modules (not just visualization)
- Wraps VTK, ITK, Matplotlib, web services, much more
- Parameter exploration
- Works with scripts
- www.vistrails.org

How did it feel?
Did it do what you want?

Traits of VisTrails

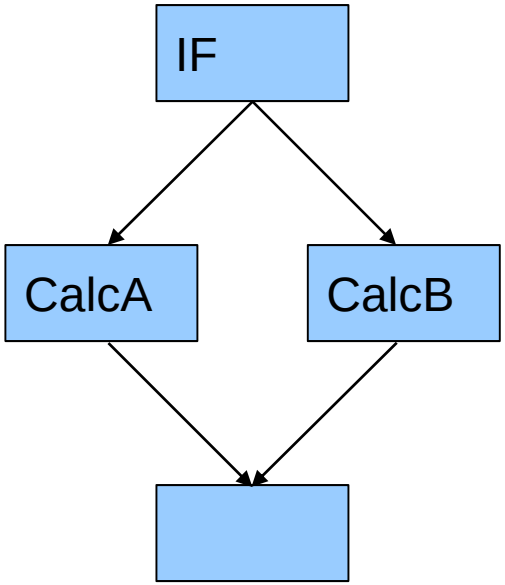
- Quick, interactive
- Steps were filters in a visualization
  - Not whole applications
  - Not lines of code
- Fine-grained control over complex activities
- Run again, history
- Architectural Components
  - UI to build and manipulate pipelines
  - Libraries of modules
  - Execution engine to run pipelines
  - Provenance recording and manipulation

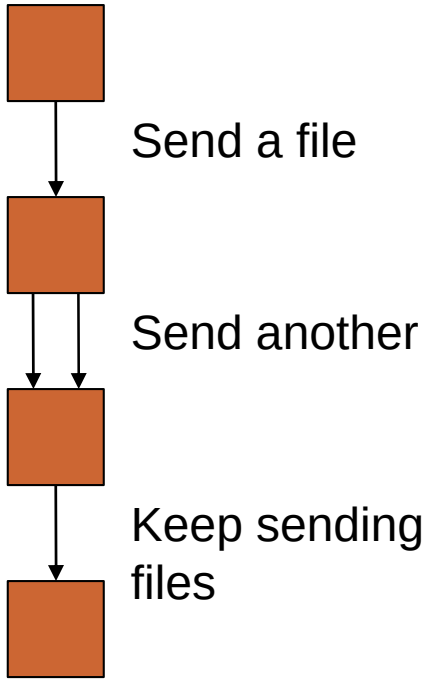
Other options?

World of Workflows (WoW)

- Interactive / batch / grid
- Fine-grained versus course modules
- Threaded, multi-process
- Level of error reporting and debugging
- Handles provenance in an automatic way

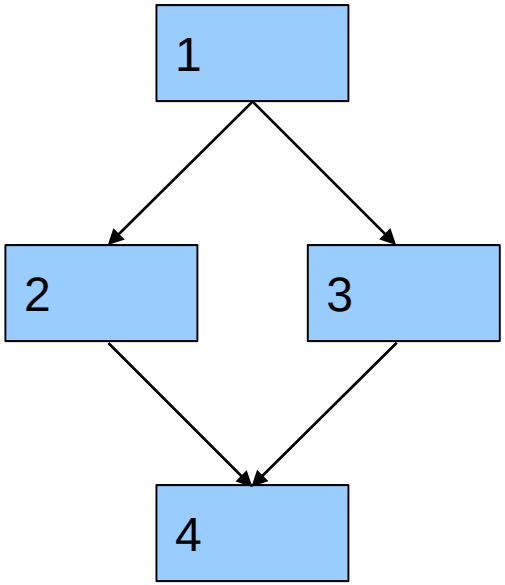
Process-oriented versus Data-driven

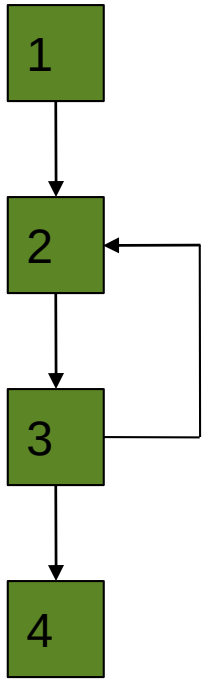


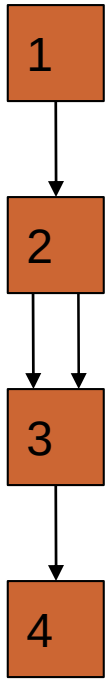


- Process-oriented can approach being a new language
- Data-driven not as dumb as it looks

What Does the Graph Mean?

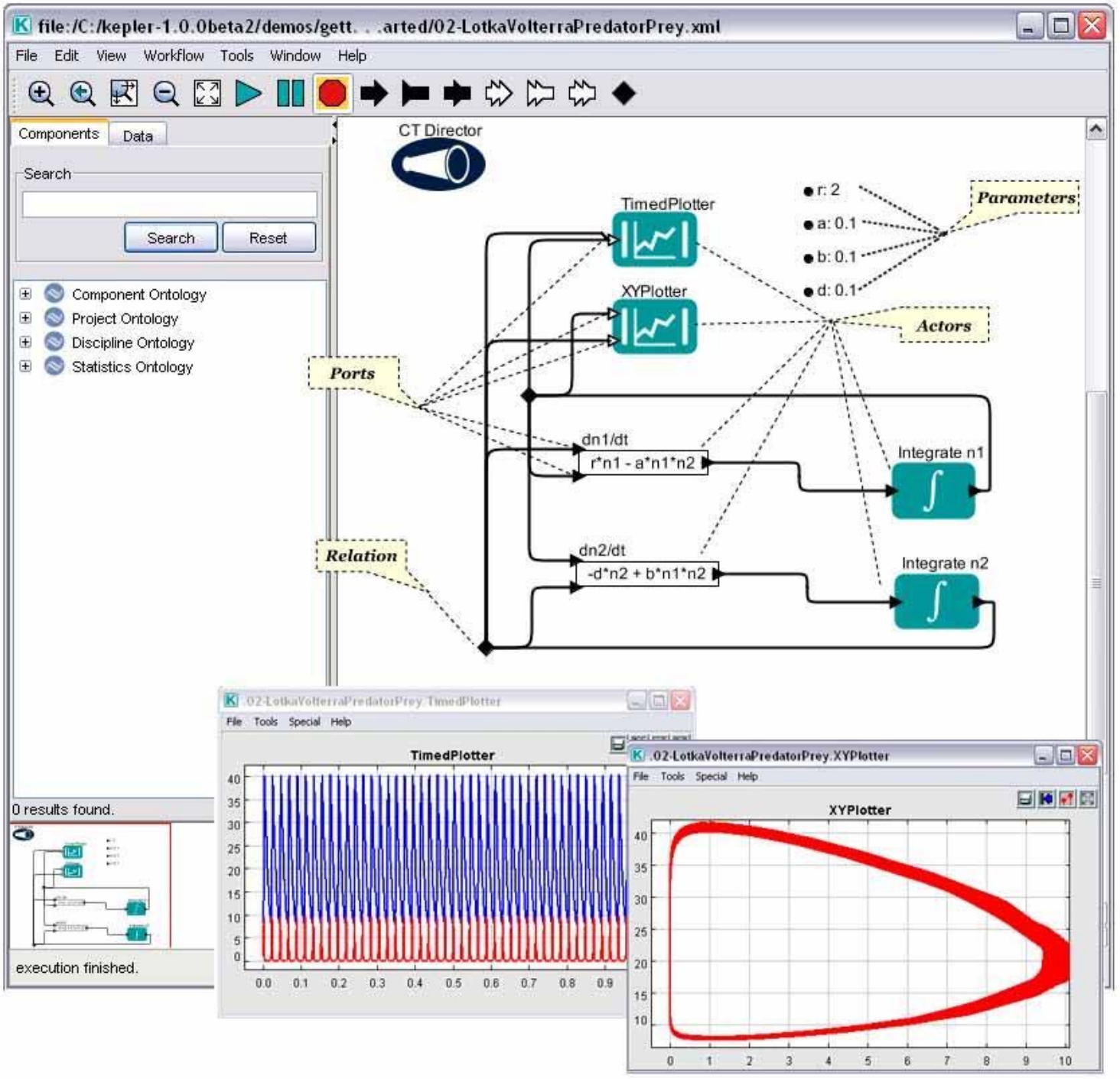






Kepler

- Java-based
- Open source, mature
- Engine runs on single machine
- Access grid tools, web services, databases
- Granularity varies from adding ints to running applications
- Do anything you can do in Java


file:/C:/kepler-1.0.0beta2/demos/gett. . .arted/02-LotkaVolterraPredatorPrey.xml
File Edit View Workflow Tools Window Help
Components Data
Search
Search Reset
Component Ontology
Project Ontology
Discipline Ontology
Statistics Ontology
0 results found.
execution finished.
CT Director
TimedPlotter
XYPlotter
r: 2
a: 0.1
b: 0.1
d: 0.1
Parameters
Actors
Ports
Relation
dn1/dt
r*n1 - a*n1*n2
Integrate n1
dn2/dt
-d*n2 + b*n1*n2
Integrate n2
02-LotkaVolterraPredatorPrey.TimedPlotter
File Tools Special Help
TimedPlotter
02-LotkaVolterraPredatorPrey.XYPlotter
File Tools Special Help
XYPlotter

Directors from Ptolemy

- Process network, driven by messages, multi-threaded
- Continuous time, explicit time dependence
- Synchronous Data Flow, for simple transformations
- Discrete event, modeling time-oriented systems, timestamped inputs
- Dynamic Data Flow, like SDF, but not pre-determined

Ideas From Kepler

- VisTrails has a remarkably similar execution model
- Domain kits of modules
- Every input/output has a type
- Components can have sub-components
- Boxes with lines among them don't always mean the same thing
  - Continuous, Discrete Time, SDF directors
  - Can you express loops, conditionals, asynchronous events?
- Whitebox
- For me, ran best under debugger for development

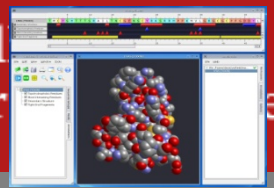
Taverna

- Bioinformatics, systems biology
- Uses ontologies specific to these areas
- myGrid team
- Virtual experiments

Components are Web Services are First Class

- Workflows can become web services
- Steering through poking web services
- Draw data from thousands of web services
- Make own modules with web services
- Ship data over separate channel for large data

University of Manchester



Client Applications

Workflow Warehouse

myExperiment Web Portal

Taverna Workbench GUI

Provenance Ontology

Provenance Warehouse

Service / Component Catalogue

Feta Information Services

Taverna Workflow Enactor

LogBook Provenance Management

Default Results

Service Ontology

3rd Party Resources (Web Services, Grid Services)

Custom Datasets

Service Management

Resources

From The Taverna Workflow System by Katherine Wolstencroft

myExperiment

- Find Workflows
- Find Files
- Share Workflows and Files
- Create and Join Groups

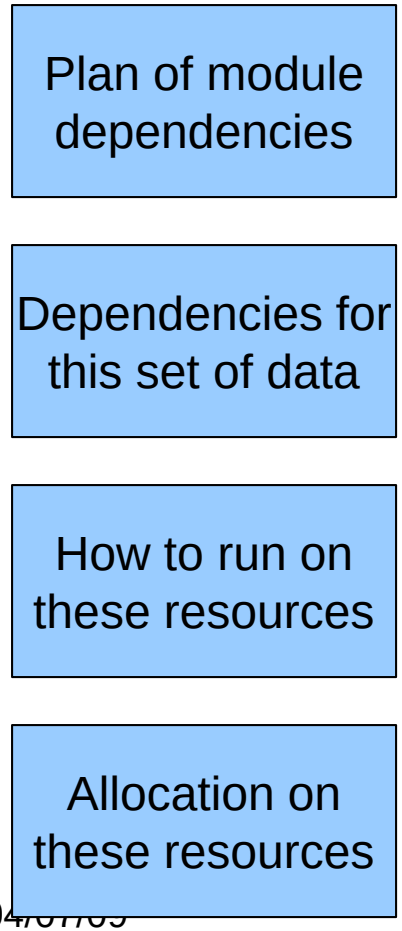
Large Scale Processing

- Pipelining, so it iterates over large datasets, giving intermediate results
- Long-running workflows
- Pass-by-reference
- Could run cluster of pipelines (different use of the word)
- Could run as cluster of web services

Psychology

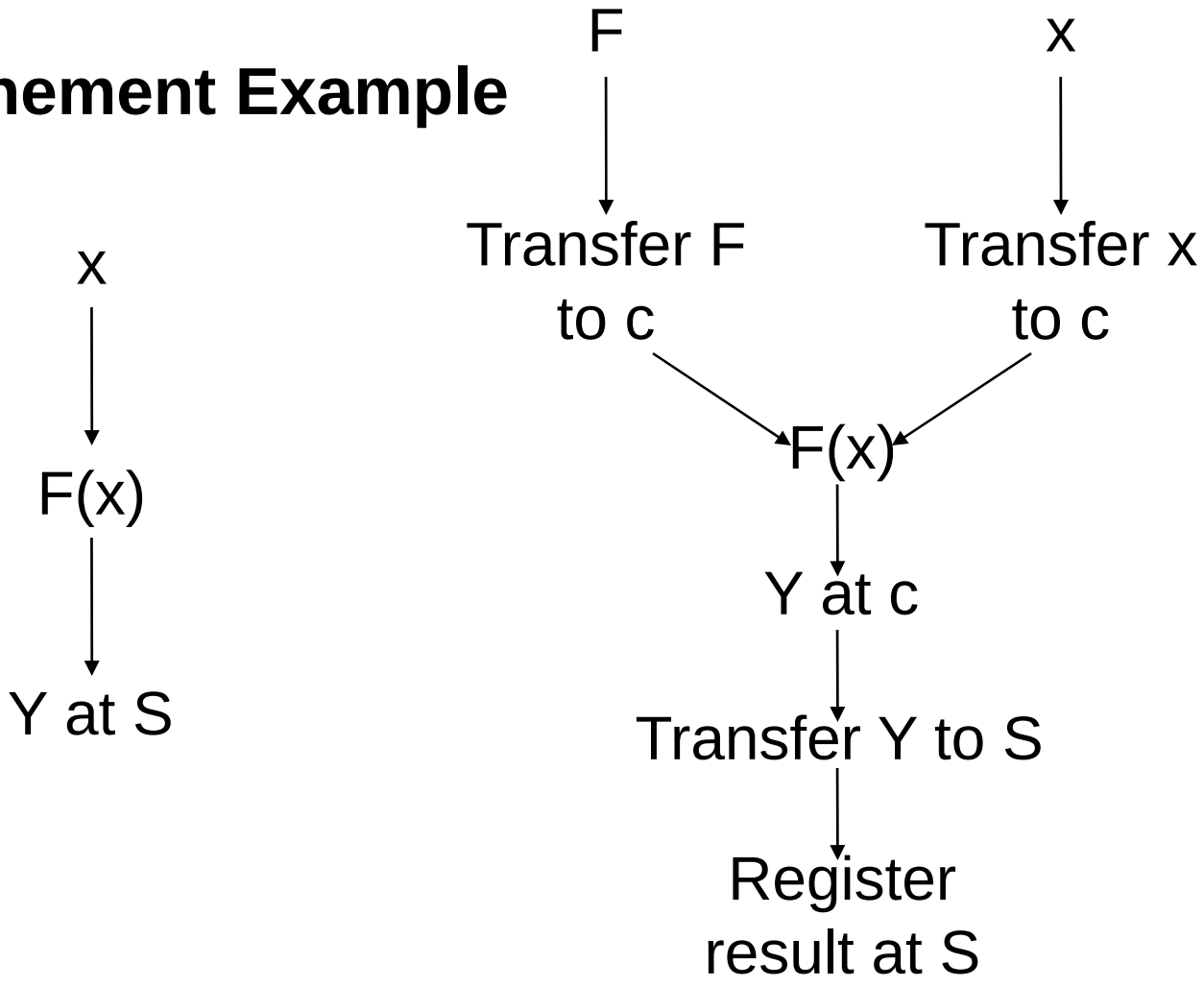
- Credited with reducing bias for running over larger datasets
- Consider psychology
  - How much do you play, investigate, when it's code versus dragging boxes
  - Do you trust models more from other sources or your own?

Wings/Pegasus



- Workflow mapping engine
- Takes into account resource efficiency, dynamic availability
- Works with scheduler, resource manager on clusters.
- Reformulates a pipeline depending on input data and available resources
- Initial workflow description is more abstract, made in DAX or Kepler

Refinement Example



From a paper by Miles, Groth, Deelman, et al in Computing in Science and Engineering.

Understanding DAX (1) from Pegasus WMS Tutorial (Guarang Mehta)

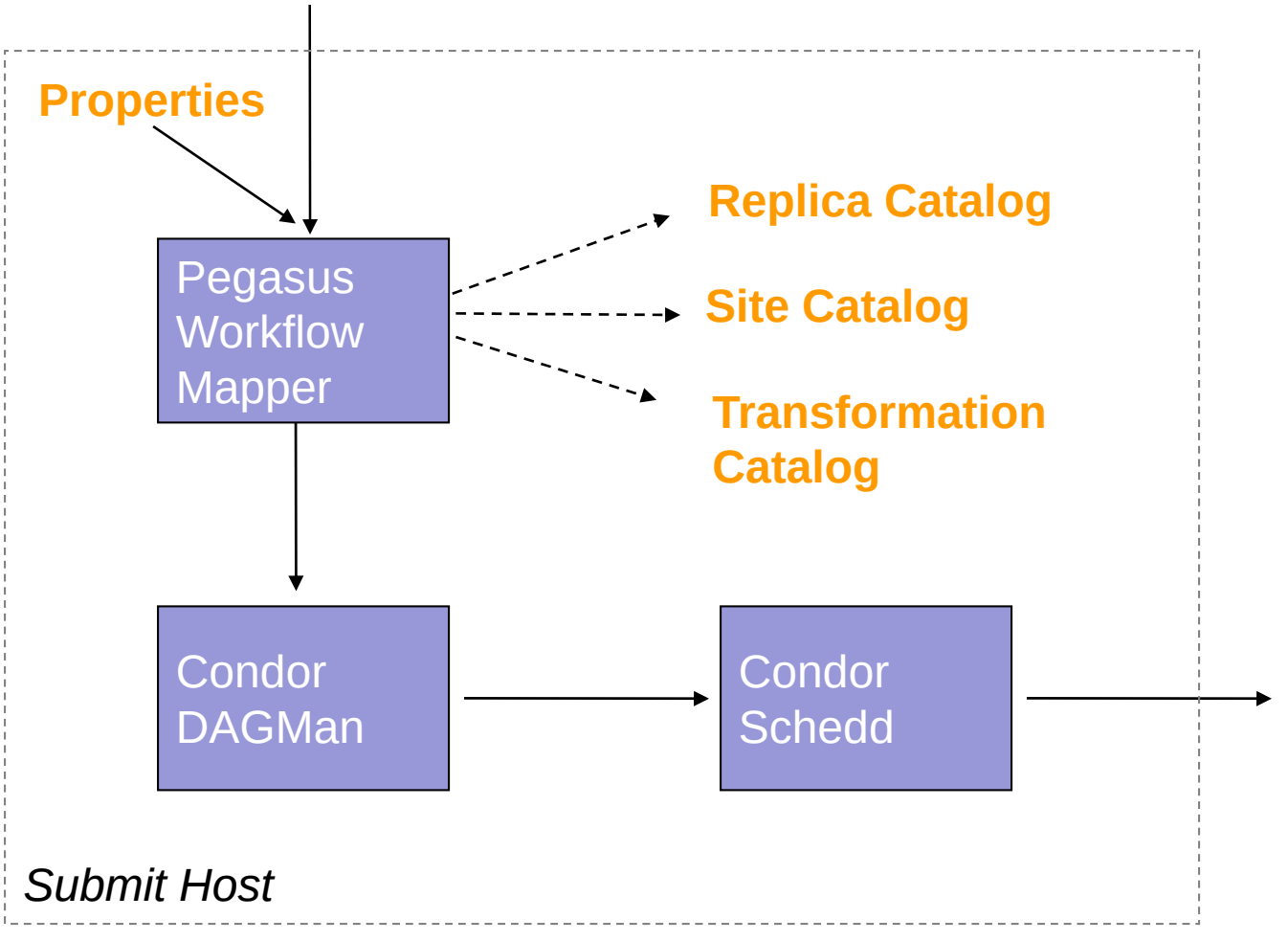
```
<!-- part 1: list of all files used (may be empty) -->
  <filename file="f.input" link="input"/>
  <filename file="f.intermediate" link="input"/>
  <filename file="f.output" link=”output"/>
  <filename file=“keg” link=“input”>
<!-- part 2: definition of all jobs (at least one) -->
  <job id="ID000001" namespace=”pegasus" name="preprocess" version="1.0" >
        <argument>-a top -T 6  -i <filename file=”f.input"/>  -o <filename file=”f.intermediate"/>
        </argument>
        <uses file=”f.input" link="input" register="false" transfer=”true"/>
        <uses file=”f.intermediate" link="output" register=”false" transfer=“false”>
<!-- specify any extra executables the job needs . Optional  -->
<uses file=“keg” link=“input” register=“false” transfer=“true” type=“executable”>
  </job>
  <job id="ID000002" namespace=”pegasus" name=”analyze" version="1.0" >
        <argument>-a top -T 6  -i <filename file=”f.intermediate"/>  -o <filename
file=”f.output"/>
        </argument>..
  </job>
<!-- part 3: list of control-flow dependencies (empty for single jobs) -->
 <child ref="ID000002">
    <parent ref="ID000001"/>
  </child>
```

(excerpted for display)

Several Steps In This

1) Eliminate redundant steps
2) Select sites for computing
3) Add processes to stage data
4) Add processes to register results
5) Bundle some actions together

Pegasus WMS



Also from Pegasus WMS Tutorial by Mehta and Deelman.

TeraGrid
Open Science Grid
Campus resources
Local machine

Pegasus WMS restructures and optimizes the workflow, provides reliability

Cornell University
Center for Advanced Computing

Pegasus to Ranger

TeraGrid condor-g Services

This content is also available in: XML RSS (click on one)

List all services types: http://info.teragrid.org/restdemo/html/tg/services

Type: condor-g

Kit: workflow.teragrid.org version: 4.0.0

Version	Name	SiteID	ResourceID	Endpoint	Support Level->Goal
6.7.18	condor-g	ncar.teragrid.org	frost.ncar.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	ncsa.teragrid.org	abe.ncsa.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	ncsa.teragrid.org	cobalt.ncsa.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	ncsa.teragrid.org	dtf.ncsa.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	ncsa.teragrid.org	lincoln.ncsa.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	ornl.teragrid.org	nstg.ornl.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	tacc.teragrid.org	lonestar.tacc.teragrid.org	embedded_in_software	production->production
6.7.18	condor-g	tacc.teragrid.org	ranger.tacc.teragrid.org	embedded_in_software	production->production
6.8.4	condor-g	iu.teragrid.org	bigred.iu.teragrid.org	embedded_in_software	production->production

Granularity

- Pegasus had files and executables.
- Taverna/Kepler/Python work on language modules
  - Can do executables
  - Some modules are small, like addition, string splitting
  - Macros/sub-workflows let you zoom out
- Code is equivalent to a graph
- Workflow reveals structure, which helps understanding

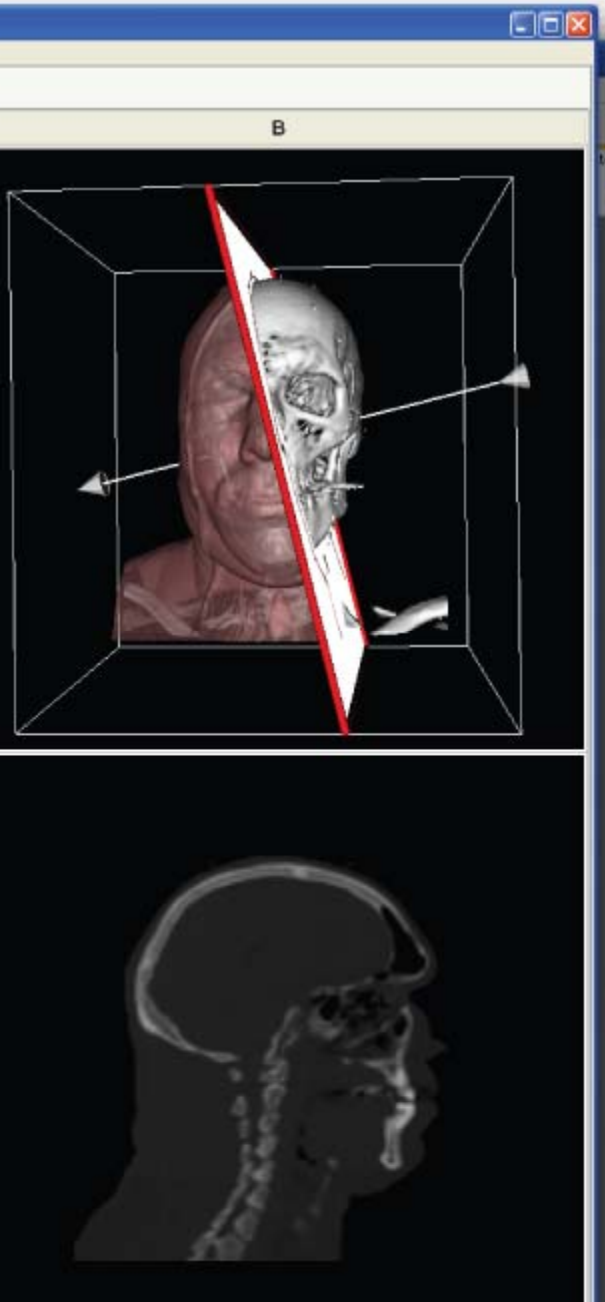
File Management in Grid Workflows

- Logical filenames in the workflow, movement by GridFTP, FTP, HTTP, NeST, SRB, dCache, CASTOR, UniTree, bbcp
- Timeouts from too many simultaneous sends
- Very difficult to get good speed without optimization for
  - Number of files
  - Size of files
  - Current network traffic
  - Destination architecture
- Stork (Condor world) schedules data placement
- Phedex for high energy physics

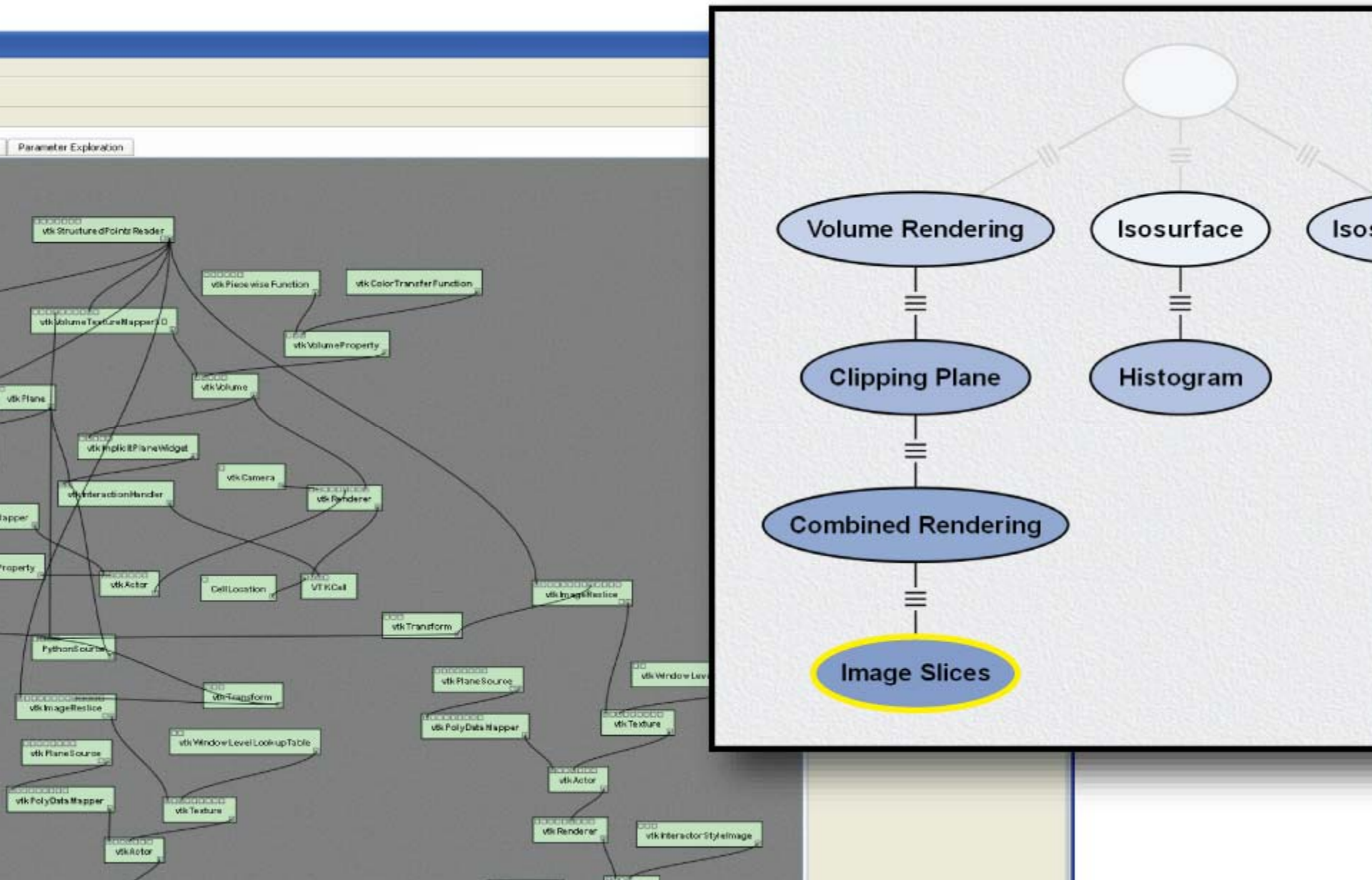
More and More Abstraction

- Handle computing at different sites, data formats, applications
- Virtual Data Language
  - Separates logical data structure from physical
  - Handles directory structure, filenames with metadata
  - Can expand into computable DAGs
- Problem-solving methods applied to domains
- Drifting into applications that serve goals you may not have.

VisTrails Provenance Example



Workflow History



Workflow History

- The history of your building of the pipeline.
- Specific to VisTrails.
- Can share with workgroup.
- Can make URIs, embed in PDF.
- New kind of inscription.
- Not about having a pipeline, executing a pipeline.
- About provenance.

Two Types of Provenance

- Prospective
  - Enough to do what you're going to do
  - Might be abstract (Pegasus), several steps from retrospective
- Retrospective
  - What was done for one dataset on one architecture
  - Processors, process times, username
- Causality is crucial
- Explicit annotation

Three General Ways to Track

- Workflow does it for you
  - Some record only prospective
  - VisTrails traces developments in provenance
- Per-process
  - Each responsible for its own
  - Karma, PASOA record it
  - Includes web services
- System-based, OS
  - Every process invocation, file open
  - LOTS of data (5 MB for one process)
  - Causality obscured

Store Provenance in Layers

- Separate algorithm, execution, data
- Details good, but separate
- Layers reduce duplication
- Not included: top layer about your work

1) Workflow template
2) Workflow instance
3) Executable workflow
4) Execution log

Remember Pegsus steps? →

1) Eliminate redundant steps
2) Select sites for computing
3) Add processes to stage data
4) Add processes to register results
5) Bundle some actions together

How to Store the Data

- Often stored in different ways
  - RDBMS
  - XML dialect
  - RDF assertions
  - SPARQL

File Metadata

- HDF5, netCDF, and others
- Can store history of each piece of data, which algorithm it came from.
- Not the same as why you ran the program. Missing causality.
- HEP measures percentage of storage and time processing metadata.

Recording Is Contagious

- Tau makes a database
- MySQL db for every compilation, every execution
- Seen studies in teaching scenarios

Process Provenance from Versioning Systems

```
#include <iostream>
static const char* g_mainVersion = "$Id$";

int main(int argc, char** argv)
{
  std::cout << g_mainVersion << std::endl;
  return 0;
}
```

- Subversion, CVS, git
- Record file versions in output
- svn propset svn:keywords “Id” main.cpp
- svn commit . ; svn update
- ./a.out: $Id: main.cpp 177 2009-04-07 03:35:44Z ajd27 $

Queries

- Want to know what work on Monday used this algorithm and made any images.
- Shared records for a collaboration.
- Query interfaces still in development.
- Still learning how to wade through data.
- Various databases aren't scaling for very large metadata sets.
- Have seen RDBMS dumped to Lucene for search.

REDUX for Windows Workflow

```
SELECT Execution.ExecutableWorkflowId, Execution.ExecutionId, Event.EventId,
    ExecutableActivity.ExecutableActivityId from Execution, Execution_Event, Event,
    ExecutableWorkflow_ExecutableActivity, ExecutableActivity, ExecutableActivity_Property_Value, Value,
    EventType as ET
where Execution.ExecutionId=Exectuion_Event.ExecutionEd
and Execution_Event.EventId=Event.EventId
and ExecutableActivity.ExecutableActivityId=ExecutableActivity_Property_Value.ExecutableActivityId
and ExecutableActivity_Property_Value.ValueId=Value.ValueId and Value.Value=Cast('-m 12' as binary)
and ((CONVERT(DECIMAL, Event.Timestamp)+0)%7)=0 and
    Execution_Event.ExecutableWorkflow_ExecutableActivityId=ExecutableWorkflow_ExecutableActivity.Exec
    utaableWorkflow_ExecutableActivityId
and ExecutableWorkflow_ExecutableActivity.ExecutableWorkflowId=Execution.ExecutableWorkflowId
and ExecutableWorkflow_ExecutableActivity.ExecutableActivityId=ExecutableActivity.ExecutableActivityId
and Event.EventTypeId=ET.EventTypeId and ET.EventTypeName='Activity Start';
```

VisTrails Query

- Wf{*}: x where x.module='AlignWarp' and x.parameter('model')='12' and (log{x}: y where y.dayOfWeek='Monday')
- Shorter? Yes. Want to type this? Still no.

User Interfaces to Provenance

- Show VisTrails?
- Ask differences between two workflows
- Parameterize workflow creation and execution
- Query-by-example
  - Select part of a workflow
  - Search for others similar
- Analogies
  - Select changes from one workflow to another
  - Apply simlar changes to a third workflow

Workflows as Inscription

- Uniform Resource Locator for a workflow
- Embed in PDF, presentation, books
- Is this a new kind of evidence?
- Does it muster support?

Appropriate Answers to Challenges

- Computation in lab, production, user spaces.
- Deeply hierarchical data in large amounts.
- Distributed collaborations.
- Two researchers, same work, different results. Months to unravel.
- Provenance likely central.
- Don't use that provenance very well yet.