Geographic Information Provenance

JAMES FREW
Donald Bren School of
Environmental Science and Management
University of California, Santa Barbara
What is Provenance?

• Information about
  - events
  - parameters
  - source data
  - responsible parties
    • (from: http://www.fgdc.gov/metadata/csdgm/02.html)

• Allows scientists to:
  - understand the origin of their results
  - repeat experiments
  - validate processes used to derive data products
    • (from: http://twiki.ipaw.info/bin/view/Challenge/WebHome)
Provenance Problems

- Capture
- Communication
The Capture Problem:

• You think like this…
The Capture Problem:

- You think like this...
- But you work like this...
#!/bin/sh

align_warp anatomy1.img reference.img warp1.warp -m 12 -q
align_warp anatomy2.img reference.img warp2.warp -m 12 -q
align_warp anatomy3.img reference.img warp3.warp -m 12 -q
align_warp anatomy4.img reference.img warp4.warp -m 12 -q

reslice warp1.warp resliced1
reslice warp2.warp resliced2
reslice warp3.warp resliced3
reslice warp4.warp resliced4

softmean atlas.hdr y null \  
resliced1.img resliced2.img resliced3.img resliced4.img

slicer atlas.hdr -x .5 atlas-x.pgm
slicer atlas.hdr -y .5 atlas-y.pgm
slicer atlas.hdr -z .5 atlas-z.pgm

convert atlas-x.pgm atlas-x.gif
convert atlas-y.pgm atlas-y.gif
convert atlas-z.pgm atlas-z.gif
The Capture Problem:

• You think like this…
• But you work like this…
• So how do you remember the connections?
Manual Provenance Capture: How?

- **Workflow system**
  - Provenance explicit in workflow graph
    - **Problem**: must learn and use workflow system

- **Wrappers**
  - Scripts contain provenance information
    - **Problem**: must create wrappers & keep them current

- **Annotation**
  - Users add post hoc metadata
    - **Problem**: (yeah right...)
Workflow Example: ArcGIS ModelBuilder
Wrapper Example: ESSW

- ESSW Database
- Perl API
- XML + SQL
- ESSW daemon
- Receive
- Ingest and Calibrate
- Navigate (Manual/Automatic)
- Sea Surface Temp (SST)
- Rectify
- SST Maps
- MySQL
- JDBC
- Java
- Perl
Annotation Example: FGDC Metadata

- <Data_Quality_Information>
  - <Logical_Consistency_Report>String</Logical_Consistency_Report>
  - <Completeness_Report>String</Completeness_Report>

- <Lineage>
  - <Source_Information>
    - <Source_Citation>
      <Originator>University of Miami/Rosenstiel School of Marine and Atmospheric Sciences</Originator>
      <Publication_Date>1991</Publication_Date>
      <Title>NOAA Advanced Very High Resolution Radiometer Multichannel Sea Surface Temperature data set</Title>
      <Other_Citation_Details>Distributed by the Distributed Active Archive Center, Jet Propulsion Laboratory, Pasadena, California. User's guide is 10 pages. The data are distributed on 9 nine-track tapes in VAX Backup format.</Other_Citation_Details>
    </Source_Citation>
    <Source_Scale_Denominator>0</Source_Scale_Denominator>
    <Type_of_Source_Media>Nine-track tape</Type_of_Source_Media>
  + <Source_Time_Period_of_Content>
    <Source_Citation_Abbreviation>AVHRR MCSST</Source_Citation_Abbreviation>
    <Source_Contribution>The source data set provides 467 weekly images of each of nine regions of the world oceans; these weekly files were averaged in the present data set to produce monthly composite images.</Source_Contribution>
  </Source_Information>
  - <Process_Step>
    <Process_Description>Calculate monthly averages and composite monthly averages. The included C-language programs sum.c and combine.c were used to calculate the monthly and weekly average sea surface temperature files. For each grid cell in the images, sum.c calculates the arithmetic average of the corresponding cell in the input files for each month or week of the year. Results are written to an set of intermediate files which are interpreted by combine.c. The program combine decodes the intermediate files written by sum and writes each average image into a new file.</Process_Description>
    <Process_Date>1993</Process_Date>
  </Process_Step>
  - <Process_Step>
    <Process_Description>Process_Description: Create GIF and PICT images of monthly and weekly averages. The C-language program mrltoppm.c converts a monthly or weekly average file into a portable pixmap. GIF and PICT images were derived from these pixmaps using the freely available PbmPlus toolkit developed by Jeff Poskanzer.</Process_Description>
    <Process_Date>1993</Process_Date>
  </Process_Step>
</Lineage>
</Data_Quality_Information>
Manual Provenance Capture Scorecard

- **Pros:**
  - Complete control over what gets recorded
  - Not tied to execution
    - You can even lie about what happened

- **Cons:**
  - Providers are customers / lack of motivation
    - Too much user interaction required
  - Must explicitly script/annotate everything
  - Scripts/annotations can drift from reality
    - You can even lie about what happened
ES3: **Automatic Provenance Capture**

**Instrumentation**
- Insert provenance capture instructions directly into science codes
  - e.g. “I just created file ‘foo’”
- Typical implementation: preprocessor/precompiler

**Overriding**
- Replace standard routines/libraries with provenance-capturing versions
  - e.g. open(…) → snoopy_open(…)
- Typical implementation: modify execution environment
  - environment variables
  - configuration files

**Passive monitoring**
- Trace program execution
  - e.g. “called open() with args = foo, bar, …”
- Typical implementation: `strace’d shell`
ES3 Provenance Architecture

Collector / Data Submission

Plugin 1
Plugin 2
... Plugin i

Annotator
Logger
Transmitter

Disk Log Files

Core / Data Storage

Web Interface

Provenance Store

User / Data Request

XML

XML / GRAPHML

Database
ES3 Provenance Architecture

• Client-side (the “Collector”)
  – plugin
    • capture real-time metadata from running process
  – Logger
    • save plugin metadata to disk
  – (optional) Annotator
    • capture existing annotation (e.g. README file)
  – Transmitter
    • format collector metadata & submit to ES3

• Server-side (the “Core”)
  – Web services
    • accept ES3 submissions/queries
  – Provenance store
    • store metadata
    • create provenance graphs
ES3 Collector: Plugins

- **IDL**
  - Hook: user startup script
  - Prepend user’s ES3 IDL directory to search path
  - Precompile user’s IDL code into ES3 IDL directory
    - Add logging code
    - Replace (some) IDL builtins with instrumented equivalents

- **bash**
  - Hook: ~/.bashrc checks ES3_ENABLE environment variable
  - Run `es3` command: traces system calls (using strace facility)
    - `es3 foo.sh` (traces `foo.sh`)
    - `es3` (traces interactive session)
ES3 Collector: Logger/Annotator

- **Logger**
  - plugin messages → XML → log file
  - Synchronous with plugin

- **Annotator**
  - Additional metadata → XML → log file
  - Use profile specified at startup
    - Text file(s)
      - Optional prepended “key:value” metadata
    - Annotation rules
      - e.g. foo.txt annotates foo.bar
    - Object characterization
      - checksum, stat(), etc.
  - Same environment as logger, but not necessarily synchronized
ES3 Collector: Transmitter

- Logger/annotator files → ES3 requests → ES3
  - Filter out irrelevant info
  - Assign UUIDs to provenance-relevant objects
  - Assemble execution traces into (sub)workflows
    - i.e. everything a particular process did

- Not necessarily same environment as logger/annotator
  - Can’t access logged/annotated objects directly
  - No independent knowledge of execution-time system state
ES3 Core

• Web services
  – Expose ES3 core functions as web request/response

• Provenance store
  – Decompose collector reports
    • Object references
    • Inter-object linkages
      – Transmitter UUIDs → primary keys
  – Reconstruct provenance graph from arbitrary start point
    • File name, process name, or UUID
    • Follow UUID references forward/backward
  – Return provenance traces in XML or GraphML
What you thought you were doing
What you actually did
Example: MODIS tile and re-project
MODIS tile and re-project: shell script and control files

mosaic.sh:
#!/bin/bash
mosaicFn="MOD09GA.A2008019.sn.005.hdf"
mrtmosaic -i tile.lis -o $mosaicFn
resample -p MRT.prm -g MRT.log

tile.lis:
MOD09GA.A2008019.h08v04.005.2008022125449.hdf
MOD09GA.A2008019.h08v05.005.2008022134646.hdf
MOD09GA.A2008019.h09v04.005.2008022151755.hdf

MRT.prm:
INPUT_FILENAME=./MOD09GA.A2008019.sn.005.hdf
SPATIAL_SUBSET_TYPE=INPUT_LAT_LONG
SPATIAL_SUBSET_UL_CORNER=(41.5000 -122.4000)
SPATIAL_SUBSET_LR_CORNER=(35.0000 -117.6000)
OUTPUT_FILENAME=MOD09GA.A2008019.sn_cal-aea.005.Refl.hdf
RESAMPLING_TYPE=NN
OUTPUT_PROJECTION_TYPE=AEA
DATUM=WGS84
OUTPUT_PROJECTION_PARAMETERS=(0.0 0.0 34.00 40.50 -120.00 0.00 0.00 \ 
-4000000.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00)
OUTPUT_PIXEL_SIZE=500
SPECTRAL_SUBSET=(0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0)
strace plugin output (edited!)

```
4810  1213121515.708913  execve("./mosaic.sh", ["mosaic.sh"], [...]) = 0
...
4810  1213121515.712317  open("/lib/libc.so.6", O_RDONLY) = 3
...
4810  1213121515.717415  execve("./mosaic.sh", O_RDONLY|O_LARGEFILE) = 3
4810  1213121520.732852  clone(...) = 4830

4830  1213121520.735487  execve("./mrtmosaic", \[
  ["mrtmosaic", ",-i", ",tile.lis", ",-o", ",MOD09GA.A2008019.sn.005.hdf"], [...]) = 0
4830  1213121520.768992  open("tile.lis", O_RDONLY) = 3
4830  1213121521.159965  open("MOD09GA.A2008019.h08v04.005.2008022125449.hdf", O_RDONLY) = 4
4830  1213121521.290125  open("MOD09GA.A2008019.h08v05.005.2008022134646.hdf", O_RDONLY) = 4
4830  1213121521.715161  open("MOD09GA.A2008019.h09v04.005.2008022151755.hdf", O_RDONLY) = 4
4830  1213121689.009340  open("tmpEi6Z73", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
4830  1213121689.071644  exit_group(0) = 0
...
4904  1213121689.301804  execve("./resample", \[
  ["resample", ",-p", ",MRT.prm", ",-g", ",MRT.log"], [...]) = 0
4904  1213121689.655122  open("./MOD09GA.A2008019.sn.005.hdf", O_RDONLY) = 3
4904  1213121689.864752  open("./MOD09GA.A2008019.sn.005.hdf", O_RDONLY) = 3
4904  1213121690.623884  open("MOD09GA.A2008019.sn_cal-aea.005.Refl.hdf", O_RDONLY) = 3
...
4904  1213121714.458947  open("MOD09GA.A2008019.sn_cal-aea.005.Refl.hdf", O_RDONLY) = 3
4904  1213121714.463607  open("./MOD09GA.A2008019.sn.005.hdf", O_RDONLY) = 3
4904  1213121714.615284  exit_group(0) = 0
```
Collector output

<init time="20080610T181514Z" stime="20080610T181155.707233Z"
pstime="20080610T181155.707233Z" pid="4783" ppid="4783" language="bash"
user="peter" hostname="localhost.localdomain">
</init>
<exec time="20080610T181515Z" routine="/mosaic.sh" pid="4810">
<arguments/>
</arguments>
<io>
<pipe read="true" id="std-in"/>
<pipe write="true" id="std-out"/>
<pipe write="true" id="std-err"/>
<file read="true">/etc/ld.so.cache</file>
<file read="true">/lib/libtermcap.so.2</file>
<file read="true">/lib/libdl.so.2</file>
<file read="true">/lib/libc.so.6</file>
<file read="true" write="true">/dev/tty</file>
<file read="true">/usr/lib/locale/locale-archive</file>
<file read="true">/proc/meminfo</file>
<file read="true">/usr/lib/gconv/gconv-modules.cache</file>
<file read="true">/home/peter/Test/ES3/RegressionTests/MODSCAG/mosaic.sh</file>
</io>
</exec>
Transmitter Output: File Object

<ES3Request type="registerFile">
  <file>
    <provenance/>
    <workflowUuid>
      b2189b33-349c-434d-bf73-3f8817dccbd5
    </workflowUuid>
    <Localfilesystem/>
      /home/peter/Test/ES3/RegressionTests/MODSCAG/mosaic.sh
    </Localfilesystem>
    <md5>23614b47b876ddee31658b1917913ed3</md5>
    <user>peter</user>
    <Timeofread>20080610T181515Z</Timeofread>
    <lastModified>20080610T181137Z</lastModified>
    <size type="b">126</size>
    <uuid>7af82a69-fa7a-4aec-abdf-eb009f5e2cab</uuid>
  </file>
</ES3Request>
Transmitter output: transformation object

<ES3Request type="storeTransformation">
  <transformation>
    <timestamp type="execution">20080610T181515Z</timestamp>
    <provenance>
      <link>
        <type>1/0</type>
        <fromUuid>7af82a69-fa7a-4aec-abdf-eb009f5e2cab</fromUuid>
      </link>
    </provenance>
    <collection>/default</collection>
    <workflowUuid>b2189b33-349c-434d-bf73-3f8817dcccdb5</workflowUuid>
    <containsWorkflowUuid>2c4310db-4949-4fabi-82e-1282432257c3</containsWorkflowUuid>
    <uuid>197dc9ee-3dbf-447b-871a-e11a0288a7ba</uuid>
    <name>./mosaic.sh</name>
  </transformation>
</ES3Request>
ES3 query: retrieve provenance

```xml
<?xml version="1.0"?>
<ES3Request type="getProvenance">
  <traversal>
    <uuidStart>d16f1729-9aa5-4cba-8fdd-5da26d9cd8eb</uuidStart>
    <direction>both</direction>
    <granularity>link</granularity>
  </traversal>
  <output>
    <format>graphml</format>
    <formatOptions>nested,yfiles</formatOptions>
    <detailLevel>full</detailLevel>
  </output>
</ES3Request>
```
ES3 provenance for `mosaic.sh` run
Another Example:
make all

CFLAGS=-O
LDFLAGS=-s -lm
OBJS=modscag.o \ 
  fileops.o mgsmix.o \ 
  geochecks.o shdnorm.o

all: modscag modsort

modscag: $(OBJS) $(CC) $(LDFLAGS) \ 
  -o $@ $(OBJS)

modsort: modsort.o $(CC) $(LDFLAGS) \ 
  -o modsort modsort.o
Forward Provenance
(from file)
Forward and Reverse Provenance (from process)
Communicating Provenance: The Open Provenance Model

- International Provenance and Annotation Workshops
  - http://www.ipaw.info/

- Provenance Challenge
  - http://twiki.ipaw.info/bin/view/Challenge

- The Open Provenance Model (v1.01)
  - http://twiki.ipaw.info/bin/view/Challenge/OPM1-01Review
Open Provenance Model: Primary Entities

• Artifact
  – Immutable piece of state, which may have a physical embodiment in a physical object, or a digital representation in a computer system.

• Process
  – Action or series of actions performed on or caused by artifacts, and resulting in new artifacts.

• Agent
  – Contextual entity acting as a catalyst of a process, enabling, facilitating, controlling, affecting its execution.
Open Provenance Model: Provenance Graph Edges

- A \rightarrow \text{used}(R) \rightarrow P
- P \leftarrow \text{wasGeneratedBy}(R) \rightarrow A
- Ag \leftarrow \text{wasControlledBy}(R) \rightarrow P
- P1 \leftarrow \text{wasTriggeredBy} \rightarrow A
- A1 \leftarrow \text{wasDerivedFrom} \rightarrow A2
- P2
Open Provenance Model: XML Serialization

http://openprovenance.org/model/example-v1.01.a.xml
Conveying Lineage Metadata

NASA

Group 1

Group 2

Group 3

Group 4