Towards Query Interoperability: PASSing PLUS

Uri Braun and Margo Seltzer
Harvard University

Adriane Chapman, Barbara Blaustein, M. David Allen and Len Seligman
MITRE
Interoperability

<table>
<thead>
<tr>
<th>Data Interoperability</th>
<th>OPM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Syntactic Interoperability</td>
<td>XML</td>
</tr>
</tbody>
</table>
Goal

Query Interoperability

Data Interoperability
  OPM

Syntactic Interoperability
  XML
Current Status

- Harmony
  - Desire to share
  - Open Provenance Model (OPM)

- Fragmentation
  - Data sets
  - Semantics
  - Visualization tools
  - Query interfaces
What is Query Interoperability?

- Given data collected on PASS
- Query results are the same
  - Query on PASS
  - Import and query on PLUS
Towards Query Interoperability

- Include *all* documented relationships among entities
- Identify *all* activities
- Resolve entity names
- Match process arguments
PASSing PLUS

- Approach interoperability by example
- Start with two systems that
  - Seek to adhere to OPM
  - Work fine on their own
- Find that they do not interoperate well
# Summary of Systems

<table>
<thead>
<tr>
<th></th>
<th>PASS</th>
<th>PLUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Team</strong></td>
<td>Harvard</td>
<td>MITRE</td>
</tr>
<tr>
<td><strong>Target</strong></td>
<td>OS level</td>
<td>Workflow</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>Linux Kernel</td>
<td>Java &amp; MySQL</td>
</tr>
<tr>
<td><strong>Recording</strong></td>
<td>System calls</td>
<td>API</td>
</tr>
</tbody>
</table>
PASS Example

```
myscript -v --ignore-case -v --case-sensitive mycfg data.in data.out
```

![Diagram of data flow with nodes: mycfg, data.in, data.out, and edges labeled 'used' and 'wasGeneratedBy']
PASS-PLUS Architecture

PASS

- convert2xml
- schema.xsd
- pass.xml
- validated pass.xml

PLUS

- Queries
- Importer
- API
Provenance Transfer

PASS

convert2xml → pass.xml → Validator → validated pass.xml

schema.xsd

Queries

PLUS

API
PASS XML: File

<provenance pnode="2" version="0">
<record type="TYPE">
  <data>FILE</data></record>
<record type="NAME">
  <data>myscript</data></record>
...

PNodes and Versions

<provenance pnode="2" version="0">
  <record type="TYPE">
    <data>FILE</data>
  </record>
  <record type="NAME">
    <data>myscript</data>
  </record>
  ...
</provenance>
Identity Information

<provenance pnode="2" version="0">
  <record type="TYPE">
    <data>FILE</data>
  </record>
  <record type="NAME">
    <data>myscript</data>
  </record>
  ...
</provenance>
Challenges with Entities

- Identification
  - `<pnode,version>` vs. uid

- Type specification
  - Version 0 only

- Version representation
Solutions for Entities

- Identification
  - `<pnode,version>` vs. uid
- Type specification
  - Version 0 only
- Version representation
PASS XML: Process

<pre>  <provenance pnode="13" version="2">
    <record type="TYPE">
      <data>PROC</data>
    </record>
    <record type="ARGV">
      <data>myscript -v --ignore-case -v --case-sensitive mycfg data.in data.out</data>
    </record>
  </provenance>
</pre>
Arguments

<provenance pnode="13" version="2">
  <record type="TYPE">
    <data>PROC</data>
  </record>
  <record type="ARGV">
    <data>myscript -v --ignore-case -v --case-sensitive mycfg data.in data.out</data>
  </record>
…
Activities

- Activity is what we are executing
  - myscript
- If we run myscript 20 times we have 20 invocations of the same activity
- PASS does not distinguish the activity
Challenges with Processes

- Encoding Arguments
- Identifying Activities
Solutions to Processes

- Encoding Arguments
- Identifying Activities

Collections
Make explicit
PASS XML: Relationship

<provenance pnode="13" version="2">
  <record type="INPUT">
    <xref pnode="2" version="1"/>
  </record>
  ...
</provenance>
Recording Relationships

- Where to records dependencies:
  - PASS descendant
  - PLUS process

- Conceptual model
  - Long living processes/files
  - Versions
Forward References

<provenance pnode="13" version="2">
  <record type="INPUT">
    <xref pnode="15" version="1"/>
  </record>
  ...
</provenance>
Unresolved References

- Refer to an entity that does not appear
- Caused by semantic assumption
  - PASS does not reify unchanged versions
Challenges with Relationships

- Entity to attach to
- Forward References
- Unresolved References
Solutions to Relationships

- Entity to attach to: Descendant
- Forward References: Sort
- Unresolved References: Disallow
Validation

PASS

convert2xml

schema.xsd

Validator

pass.xml

validated pass.xml

Importer

Queries

PLUS

PASSing PLUS

Towards Query Interoperability: PASSing PLUS

TAPP '10
Challenges with Validation

- OPM validation is via XML schema definition (xsd)
- No way to enforce:
  - No self-loops
  - Acyclic
  - Other problems we just discussed
Solutions to Validation

- Need something stronger than xsd
- Schematron or a similar tool could enforce these rules
## Summary of Challenges & Solutions

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Dictionaries</td>
</tr>
<tr>
<td>Type Specification</td>
<td>Attach to all versions</td>
</tr>
<tr>
<td>Versions &amp; Arguments</td>
<td>Collections</td>
</tr>
<tr>
<td>Activities</td>
<td>Make Explicit</td>
</tr>
<tr>
<td>Relationship Recording</td>
<td>Descendant</td>
</tr>
<tr>
<td>Forward References</td>
<td>Sort</td>
</tr>
<tr>
<td>Unresolved References</td>
<td>Disallow</td>
</tr>
<tr>
<td>Validation</td>
<td>Schematron</td>
</tr>
</tbody>
</table>
Suggestions

■ Concepts
  ● Dictionaries
  ● Collections

■ Constraints
  ● Sort
  ● No Unresolved

■ Validation
  ● Schematron
Conclusions

- Our goal should be query interoperability
  - Enable sharing of tools
- We have more work to do on semantics
  - Concepts
  - Constraints
  - Validation
Towards Query Interoperability:
PASSing PLUS

Uri Braun and Margo Seltzer
Harvard University

Adriane Chapman, Barbara Blaustein,
M. David Allen and Len Seligman
MITRE