Samuel Grayson, Reed Milewicz, Joshua Teves, Daniel S. Katz, Darko Marinov

PROBLEM: ARTIFACT EVALUATION

- Author has to write plain-English description
- Artifact evaluator (AE) tries to follow
- Author: You didn’t follow the instructions exactly!
- AE: You’re instructions were ambiguous/unclear.
- Future user: Author and AE got it to work. How exactly?

PROBLEM: LARGE SCALE RE-EXECUTION STUDIES

- Repeatability in Computer Systems by Collberg and Proebsting
  - Large manual effort → how many research codes still run?
- How do we reproduce the reproduction?
  - Instructions here are not machine-readable
- Authors: it would have worked; you just didn't invoke the right commands! here

INSIGHTS

- Everyone figures out how to reproduce by themselves
  - Correct documentation requires manual effort
  - Often absent, out-of-date, incomplete, or ambiguous
- Want a *machine-readable* way to *capture* and *share* reproducibility instructions
  - Authors → with AE, readers
  - Re-executors → other re-executors, readers

INSIGHTS

- Instructions needed to reproduce = retrospective provenance
- Most important is commands/arguments
- Commands define software environment
  - Other approaches (CDE) too expensive (storage, perf)
- Imperfect data still better starting point
- This data is *automatically* collectable

BENEFITS TO AUTHORS

- “Pushbutton” artifact evaluation
- Automatic uncertainty quantification
- Regression tests/CI

RESEARCH OPPORTUNITIES

Makes research software studyable by software engineering researchers

- Automatic repair studies
- Reproducibility assessment
- Provenance overhead
- Performance impact

EXAMPLE

```
<rdf:RDF>
  <process rdf:about="#make">
    <command>make all</command>
  </process>
  <process rdf:about="#run" depends-on="#make">
    <command>./simulate</command>
    <prov:generated>
      <doco:figure>
        <rdf:Description>
          <dc:title>Figure 2b</dc:title>
          <dc:isPartOf rdf:resource=
            "https://doi.org/10.1234/123456789" />
        </rdf:Description>
      </doco:figure>
    </prov:generated>
  </process>
</rdf:RDF>
```

SPECIFICATION REQUIREMENTS

- Not a workflow engine, but can invoke one
- Decentralized dataset
  - Store with code repo or third party repo
  - Can be uploaded by authors, users, or re-executors (not just authors!)
- Shell is lingua franca
  - Better semantics if we recognize shell command
- Should support optional data that is more complex to collect
  - Automatic: Files read/written (1-15% overhead)
  - Manual: Input/output types

LINKABLE ONTOLOGIES AND DATA PROVIDERS

- wf4ever
- Nepomuk File Ontology (NFO)
- Document Component Ontology (DoCO)
- Nanopublications
- Description Of A Project (DOAP)
- Transitive Credit/CRediT
- ORCID
- Datacite Ontology

WHY NOT ___?

- Workflow engine
  - Which one? Support all?
- Docker/Nix/Guix
  - Not all experiments can be Dockerized or Nix-ified
  - What volume/flags?
  - Oriented more towards build-phase
- Scripts or CI
  - Could leverage!
  - Machine readable way to link script → linked data?
  - Easier to go the other way, linked data → scripts

WHY NOT ___?

- CDE/SciUnit/ReproZip
  - Could leverage!
  - Too expensive, few users
- Sumatra
  - Could leverage!
  - Sharable data in interoperable form

WANTED: A COMMUNITY EFFORT

- Interested stakeholders
  - Computational scientists
  - Research software engineers/researchers
  - Provenance researchers
- Exemplars
- https://github.com/charmoniumQ/execution-description