WANTED: STANDARDS FOR AUTOMATIC REPRODUCIBILITY OF COMPUTATIONAL EXPERIMENTS

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

PROBLEM: ARTIFACT EVALUATION

- Author has to write plain-English description
- AE has to spend effort interpreting them
- 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
  - How many research codes could we still run?
- How do we reproduce the reproduction? here
- Authors: it would have worked; you just didn’t invoke the right commands! (BarowyCB here)

INSIGHTS

- Everyone figures out how to reproduce by themselves.
  - Documentation is hard to write
  - Documentation, even if written, can be ambiguous
- Want a *machine-readable* way to *share* reproducibility instructions
  - Authors → with AE, readers
  - Re-executors → other re-executors

INSIGHTS

- Instructions to share = retrospective provenance
- Most important is commands/arguments
- Even imperfect data would be better starting point for automatic repair
- 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

- Reproducibility assessment
- Provenance overhead
- Automatic repair studies
- 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/10.1234/123456
        </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!)
  - Clients search code repo and third party repos
- With enough data, automatically re-executable
- Shell is lingua franca
  - Better semantics if we recognize shell command

RELATED ONTOLOGIES

- wf4ever/wfdesc
- DoCO
- Nanopublications
- DOAP
- Transitive CRediT

OTHER DATA

- The format should support optional data that is more complex to collect
  - Automatic: Files read/written (1-15% overhead)
  - Manual: Input/output types
  - Manual: Classify parameters as {calibration, fidelity, random seed}
  - Manual: Link to publication

ALTERNATIVES

- `make all`
- `docker build .`
- Workflow engine

WANTED: A COMMUNITY EFFORT

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

FEEDBACK/CHANGE

- introduce provenance earlier
- AE -> Artifact evaluators
- (Proebsting and Collberg) title, point of paper
- Move optionally collected data
- Capture install commands or not?
- Automatically collectible appears twice
- Justify why not use make
- Title slide should be picture
- Change title of “Specification”
- resource bleed off scren
- Shell command in linked data