A Quantitative Study on the Re-executability of Publicly Shared Scientific Workflows

Rudolf Mayer, Andreas Rauber

SBA Research, Austria
Outline

• Introduction & Motivation
• Data Set & Experiment Setup
• Workflow processors & characteristics
• Results & Recommendations
• Conclusions
Introduction & Motivation

• Repeatability & Reproducibility cornerstones
  – Computational/eScience shouldn’t be an exception ...

• Workflows have become a popular mean to share and publish scientific experiments
  – Describe & formalise the steps of an experiment
  – Bundle scripts & code with the workflow definition
  – (More) platform independent
  – Facilitate & should enable repeatability
Introduction & Motivation

• Are workflows really easily repeatable?
  – At least **re-executable**?

• Prior studies on re-execution / reproducibility
  – In-depth analysis, but small in size: a handful of example workflows

• Need for a larger analysis to see patterns
Introduction & Motivation

- Our study: large scale quantitative analysis
- Obtain workflows from a public platform dedicated to sharing scientific work – Published by authors ➔ should be „better quality“
  - Try to re-execute the workflows
    - Record data on the reasons for failure along
- Analyse the most common reasons for failures
- Recommendations for enabling better re-executability
Outline

• Introduction & Motivation
• Data Set & Experiment Setup
• Workflow processors & characteristics
• Results & Recommendations
• Conclusions
Data Set

- Source: myExperiment.com (as of March 2015)
  - Available since 2007
  - Data often published by original researchers
  - 300-400 workflows uploaded per year
Data Set

- Obtained workflow definition & meta-data
  - using myExperiment REST API

- As of March 2015:
  - ~ 2,700 workflows shared
  - Small number (92) is private
  - 40 different workflow engines
  - Majority for Taverna 2 engine

<table>
<thead>
<tr>
<th>Workflow Engine</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taverna 2</td>
<td>54.7</td>
</tr>
<tr>
<td>Taverna 1</td>
<td>20.9</td>
</tr>
<tr>
<td>RapidMiner</td>
<td>10.0</td>
</tr>
<tr>
<td>Galaxy</td>
<td>2.0</td>
</tr>
<tr>
<td>Others</td>
<td>12.4</td>
</tr>
</tbody>
</table>
Experiment Setup

• Focus on Taverna 2 workflows  
  – Account for 55% of the data set  
  – Taverna API to analyse & run in batch mode  
  – Final data set: **1,443 workflows**

• Static analysis: collect information on types of workflow processing elements

• Re-execution: automatically execute workflows  
  – Utilise example values for workflow input parameters  
  – Collect execution status, logs & provenance for each workflow
Outline

• Introduction & Motivation
• Data Set & Experiment Setup
• Workflow processors & characteristics
• Results & Recommendations
• Conclusions
Workflow Processors & Characteristics

• Types of processing elements in Taverna
  
  – Web Services
    • WSDL & REST
  
  – Beanshell
    • Execution of Java Code
  
  – Rshell
    • Execution of R Code on RServer Instance
  
  – Tool
    • Any command available on a local or remote system
Workflow Processors & Characteristics

- Types of processing elements in Taverna
  - LocalWorker
    - Beanshell with pre-defined functionality
    - E.g. Fetching contents of a URL, Base64 Encoding, ...
  - Xpath queries
    - "Trivial" processors
      - XML Processing
        - Extracting single values / combining to document
      - String Constants

Workflow Processors & Characteristics

• Processor usage in workflows
  – ~12 processor steps per workflow

<table>
<thead>
<tr>
<th>Processor</th>
<th>%</th>
<th>% nonTrivial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trivial Processors</td>
<td>&gt; 40</td>
<td></td>
</tr>
<tr>
<td>LocalWorker</td>
<td>18.38</td>
<td>30.4</td>
</tr>
<tr>
<td>BeanShell</td>
<td>15.18</td>
<td>25.1</td>
</tr>
<tr>
<td>WSDL</td>
<td>4.03</td>
<td>13.2</td>
</tr>
<tr>
<td>Tool</td>
<td>3.21</td>
<td>6.7</td>
</tr>
<tr>
<td>REST</td>
<td>1.93</td>
<td>5.3</td>
</tr>
<tr>
<td>RShell</td>
<td>1.92</td>
<td>3.2</td>
</tr>
<tr>
<td>XPath</td>
<td>1.78</td>
<td>3.2</td>
</tr>
</tbody>
</table>
Workflow Processors & Characteristics

- Web Services
  - WSDL & REST

- Vulnerabilities
  - Address not reachable (private INet address)
  - Service not available anymore
  - Service requires authentication
  - Method removed / interface changed

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WSDL</td>
<td>411</td>
<td>30</td>
</tr>
<tr>
<td>REST</td>
<td>172</td>
<td>12.6</td>
</tr>
<tr>
<td>Soaplab</td>
<td>10.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Any web service</td>
<td>599</td>
<td>42.8</td>
</tr>
</tbody>
</table>
Workflow Processors & Characteristics

• Beanshell
  – Execution of Java code

• Vulnerabilities
  – Java version executing the WF is not compatible with code version requirements
  – Dependency not available (not packed with WF)
    • Dependencies in wrong version
  – Beanshell accesses functionality outside the engine

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beanshells</td>
<td>717</td>
<td>49.7</td>
</tr>
<tr>
<td>with Dependencies</td>
<td>76</td>
<td>5.3</td>
</tr>
</tbody>
</table>
Workflow Processors & Characteristics

- **RShell**
  - Execution of R code on RServer instance
    - 90 WFs, 337 processors
    - 335 addresses *local*

<table>
<thead>
<tr>
<th>WFs with Rshell</th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90</td>
<td>6.2</td>
</tr>
</tbody>
</table>

- **Vulnerabilities**
  - Address of RServer not reachable / available
  - Authentication data missing/incorrect
  - Different version of R runtime
  - Custom R package not available / wrong version
Workflow Processors & Characteristics

- Tools: local & remote
  - Execution of arbitrary binaries
    - E.g. Image processing
    - Also: Perl / Python scripts

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local tools</td>
<td>240</td>
<td>16.6</td>
</tr>
<tr>
<td>Remote tools</td>
<td>20</td>
<td>0.8</td>
</tr>
</tbody>
</table>

- Vulnerabilities
  - Location not reachable / available
    - Authentication data missing / wrong
  - Tool is not available, or can not be found
  - Wrong version of tool installed
Workflow Processors & Characteristics

• Workflow inputs
  – Runtime parameters for workflow
    • E.g. a data source

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>No input ports</td>
<td>345</td>
<td>23.9</td>
</tr>
<tr>
<td>No/some example values</td>
<td>429/97</td>
<td>29.7/6.7</td>
</tr>
<tr>
<td>All example values</td>
<td>572</td>
<td>39.6</td>
</tr>
<tr>
<td>WF that can be run</td>
<td>917</td>
<td>63.5</td>
</tr>
</tbody>
</table>

• Vulnerabilities
  – No example values provided
  – Example values not correctly formatted
  – Example values not valid anymore
Outline

• Introduction & Motivation
• Data Set & Experiment Setup
• Workflow processors & characteristics
• Results & Recommendations
• Conclusions
Re-Execution Results

- Data set for re-execution

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original Data Set</strong></td>
<td>1443</td>
</tr>
<tr>
<td>- Missing input values</td>
<td>526</td>
</tr>
<tr>
<td>- Disabled processors (WSDL services)</td>
<td>180</td>
</tr>
<tr>
<td>- Not executable in test environment</td>
<td>6</td>
</tr>
<tr>
<td><strong>Final Data Set</strong></td>
<td>731</td>
</tr>
</tbody>
</table>
Re-Execution Results

- Execution times for workflows (in seconds)
Re-Execution Results

• Execution results

<table>
<thead>
<tr>
<th></th>
<th># WFs</th>
<th>% WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not terminated &gt;48hours</td>
<td>6</td>
<td>0.8</td>
</tr>
<tr>
<td>Execution failed</td>
<td>384</td>
<td>52.5</td>
</tr>
<tr>
<td>Execution successful</td>
<td>341</td>
<td>46.6</td>
</tr>
</tbody>
</table>

– Majority of workflows fails
– No analysis on correctness of results
– Considering full data set: only 29.2 % are successfully executed
# Re-Execution Results

- Reasons for failures
  - Manual analysis of output logs

<table>
<thead>
<tr>
<th>Reason for Failure</th>
<th># WFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>REST service unavailable</td>
<td>4</td>
</tr>
<tr>
<td>REST service unauthenticated</td>
<td>5</td>
</tr>
<tr>
<td>Other unauthenticated (WSDL, Tool)</td>
<td>40</td>
</tr>
<tr>
<td>Missing Resources (File, URL)</td>
<td>14</td>
</tr>
<tr>
<td>Local/remote Tool unavailable</td>
<td>19</td>
</tr>
</tbody>
</table>
Recommendations

- Capabilities of Workflow engines
  - Provide more scripting languages (e.g. Activiti engine) ➔ will reduce number of tool invocations

- Increase expressiveness of WF definition
  - Integrate some aspects from e.g. Research Objects or Context Model [1], e.g. dependency definition ➔ will reduce issues with missing/wrong dependencies

Recommendations

• Tool invocation
  – Dependency definition
  – Allow identification & alternatives for OS-dependent code
  ➡ Will reduce number of wrong OS / missing dependencies

• External dependencies
  – Archive & publish utilised WSDL together with workflow ➡ will enable easier substation of service
Recommendations

• Quality check & monitoring
  – Integrated into sharing platform (e.g. myExperiment), similar to WF4Ever project

• Services that check
  – Completeness of definition
    • E.g. all input example parameters are provided
  – Required Java libraries included

• Monitoring for external services to also include REST, RSHell, SSH invocations, ...
Recommendations

- Many aspects can be automated
  - Automatic capturing of process dependencies and required resources [2]

- Verification of **correct** execution to be addressed separately [3]


Outline

- Introduction & Motivation
- Data Set & Experiment Setup
- Workflow processors & characteristics
- Results & Recommendations
- Conclusions
Conclusions & Future work

• Workflows should foster reproducibility
• Many shared workflows fail at re-execution
  – “Trivial” aspects (e.g. lack of example input data)
  – Publishing/packaging resources, external dependencies
  – Local dependencies (tool executions)

• Remedies: better documentation, dependency management, monitoring of external services
• Via improvements in workflow engines, workflow definition & monitoring
Thank you!

Rudolf Mayer, SBA Research, Austria
rmayer@sba-research.org

http://www.sba-research.org/digital-preservation
http://ifs.tuwien.ac.at/dp/process