COMMON WORKFLOW LANGUAGE

researchobject.org

bioexcel
Center of Excellence for Computational Biomolecular Research

elixir

Capturing interoperable reproducible workflows with Common Workflow Language

Stian Soiland-Reyes, Farah Zaib Khan, Richard O. Sinnott, Andrew Lonie, Michael R Crusoe, Carole Goble

THE UNIVERSITY OF MELBOURNE

MANCHESTER 1824
The University of Manchester

@soilandreyes

https://orcid.org/0000-0001-9842-9718

https://slides.com/soilandreyes/

Workshop for Research Objects (RO2018),
IEEE eScience 2008, Amsterdam
2018-10-29

This work has been done as part of the BioExcel CoE (www.bioexcel.eu), a project funded by the European Union contract H2020-EINFRA-2015-1-675728.

Why use workflows?

Automation

- Automate computational aspects
- Repetitive pipelines, sweep campaigns

Scaling – compute cycles

- Make use of computational infrastructure & handle large data

Abstraction – people cycles

- Shield complexity and incompatibilities
- Report, re-use, evolve, share, compare
- Repeat – Tweak - Repeat
- First class commodities

Provenance - reporting

- Capture, report and utilize log and data lineage auto-documentation
- Traceable evolution, audit, transparency
- Compare

Findable
Accessible
Interoperable
Reusable
(Reproducible)

Three broad categories

https://doi.org/10.1186/s12859-017-1747-0

Domain-specific pre built pipelines

GUI based integrative workbenches

Standardized declarative approaches

Cpipe

bcbio

OmicsPIPE

Apache Taverna

Galaxy PROJECT

Open for Innovation KNIME

Kepler Your Science. Enabled.

Pegasus

COMMON WORKFLOW LANGUAGE

{wdl}

@farahzk03

https://slides.com/farahzkhan/cwlprov

https://s.apache.org/existing-workflow-systems

Existing Workflow systems

Michael R. Crusoe edited this page 14 days ago · 186 revisions

Computational Data Analysis Workflow Systems

Permalink: https://s.apache.org/existing-workflow-systems

An incomplete list

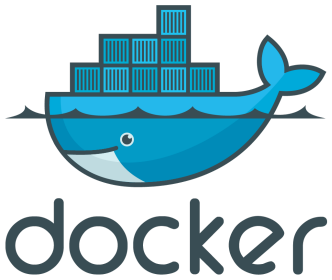
Please add new entries at the bottom.

See also: https://github.com/pditommaso/awesome-pipeline

1. Arvados http://arvados.org
2. Taverna http://www.taverna.org.uk/
3. Galaxy http://galaxyproject.org/
4. SHIWA https://www.shiwa-workflow.eu/
5. Oozie https://oozie.apache.org/
6. DNANexus https://wiki.dnanexus.com/API-Specification-v1.0.0/IO-and-Run-Specifications# https://wiki.dnanexus.com/API-Specification-v1.0.0/Workflows-and-Analyses#
7. BioDT http://www.biodatomics.com/
8. Agave http://agaveapi.co/live-docs/
9. DiscoveryEnvironment http://www.iplantcollaborative.org/ci/discovery-environment
10. Wings http://www.wings-workflows.org/
11. Knime https://www.knime.org/

208. S4M https://bitbucket.org/uqokorn/s4m_base/wiki/Home
209. Loom http://med.stanford.edu/gbsc/loom.html https://github.com/StanfordBioinformatics/loom http://loom.readthedocs.io/en/latest/templates.html
210. Watchdog https://doi.org/10.1186/s12859-018-2107-4 https://github.com/klugem/watchdog
211. phpflo https://github.com/phpflo/phpflo
212. BASTet: Berkeley Analysis and Storage Toolkit https://openmsi.nersc.gov/openmsi/client/bastet.html https://biorack.github.io/BASTet/ https://doi.org/10.1109/TVCG.2017.2744479
213. Tavaxy: Pattern based workflow system for the bioinformatics domain http://www.tavaxy.org/
214. Ginflow: Decentralised adaptive workflow engine https://ginflow.inria.fr/
215. SciApps: A cloud-based platform for reproducible bioinformatics workflows https://doi.org/10.1093/bioinformatics/bty439 https://www.sciapps.org/

COMMON WORKFLOW LANGUAGE



Implementations

Software	Description
cwltool	Reference implementation of CWL
Arvados	Distributed computing platform for data analysis on massive data sets. Using CWL on Arvados
Toil	Toil is a workflow engine entirely written in Python.
Apache Taverna	Domain-independent Workflow Management System
Galaxy	Web-based platform for data intensive biomedical research.
AWE	Workflow and resource management system for bioinformatics data analysis.
Funnel	Use Google Genomics Pipeline API with CWL
Rabix Bunny	Reproducible Analyses for Bioinformatics

```
cwlVersion: v1.0
class: Workflow
inputs:
  inp: File
  ex: string

outputs:
  classout:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: inp
      extractfile: ex
    out: [example_out]

  compile:
    run: arguments.cwl
    in:
      src: untar/example_out
    out: [classfile]
```

http://www.commonwl.org/

https://w3id.org/cwl/v1.0/

Common Workflow Language (CWL) Workflow Description, v1.0

This version:

- https://w3id.org/cwl/v1.0/

Current version:

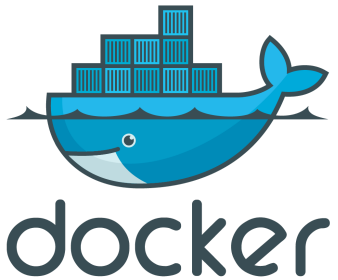
- https://w3id.org/cwl/

Authors:

- Peter Amstutz peter.amstutz@curoverse.com, Arvados Project, Curoverse
- Michael R. Crusoe michael.crusoe@gmail.com, Common Workflow Language project
- Nebojša Tijanić nebojsa.tijanic@sbgenomics.com, Seven Bridges Genomics

Contributors:

- Brad Chapman bchapman@hsph.harvard.edu, Harvard Chan School of Public Health
- John Chilton jmchilton@gmail.com, Galaxy Project, Pennsylvania State University
- Michael Heuer heuermh@berkeley.edu,,UC Berkeley AMPLab
- Andrey Kartashov Andrey.Kartashov@cchmc.org, Cincinnati Children's Hospital
- Dan Leehr dan.leehr@duke.edu, Duke University
- Hervé Ménager herve.menager@gmail.com, Institut Pasteur
- Maya Nedeljkovich maja.nedeljkovic@sbgenomics.com, Seven Bridges Genomics
- Matt Scales mscales@icr.ac.uk, Institute of Cancer Research, London
- Stian Soiland-Reyes soiland-reyes@cs.manchester.ac.uk, University of Manchester
- Luka Stojanovic luka.stojanovic@sbgenomics.com, Seven Bridges Genomics



Abstract

One way to define a workflow is: an analysis task represented by a directed graph describing a sequence of operations that transform an input data set to output. This specification defines the Common Workflow Language (CWL) Workflow description, a vendor-neutral standard for representing workflows intended to be portable across a variety of computing platforms.

Over 5000 CWL Descriptions on GitHub

cwlVersion / Pull requests Issues Marketplace Explore

Repositories
Code 5K
Commits 80
Issues 267
Marketplace
Topics
Wikis 8
Users

Languages

Common Workflow Language ✕
YAML 256
Python 178

5,706 code results

Sort: **Best match**

Common Workflow Language

IBMSpectrumComputing/cwlexec – cwl_no_class.cwl

Showing the top match Last indexed on Apr 18

src/test/resources/definitions/cwl_no_class.cwl

```
1  cwlVersion: v1.0
```

Common Workflow Language

IBMSpectrumComputing/cwlexec – cwl_unsupported_version.cwl

Showing the top match Last indexed on Apr 18

```
1  cwlVersion: v2.0
2  class: CommandLineTool
```

Common Workflow Language

rekols/recat-emacs – cwl.cwl

Showing the top match Last indexed on May 3

```
1  cwlVersion: v1.0
```

http://www.commonwl.org/user_guide/

Common Workflow Language User Guide

Hello!

This guide will introduce you to writing tool wrappers and workflows using the Common Workflow Language (CWL). This guide describes the current stable specific 1.0.

Note: This document is a work in progress. Not all features are covered, yet.

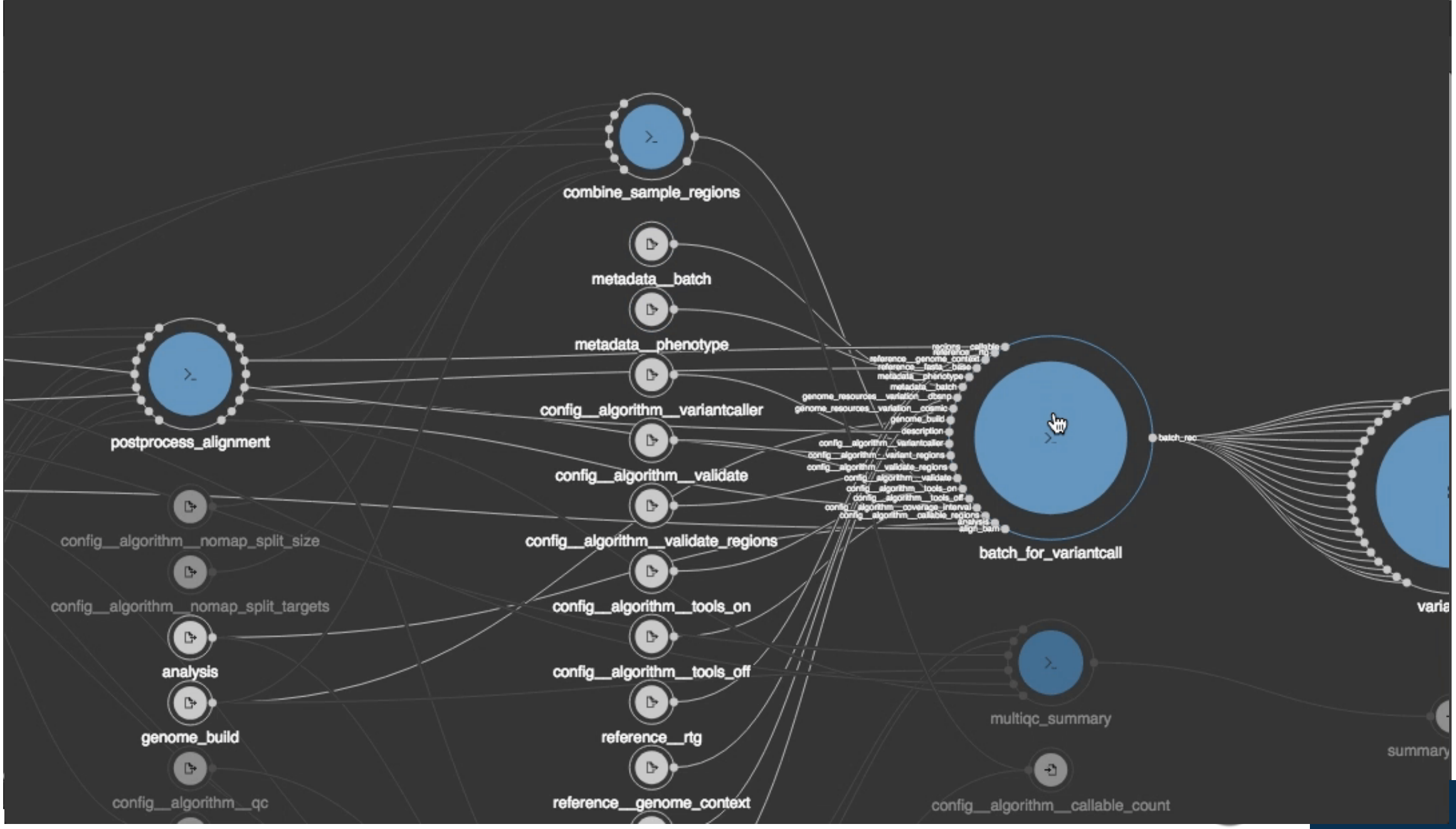
Prerequisites

A text editor

A CWL runner. It is recommended to start with the reference implementation. The full list of CWL runners is on the project homepage.

Schedule

	Setup	Download files required for the lesson
00:00	1. Introduction	What is Common Workflow Language? Why might I want to learn to use CWL?
00:00	2. First Example	How do I wrap a simple command line tool?
00:05	3. Essential Input Parameters	How do I describe inputs to a command? How do I specify the order in which inputs appear in a command?
00:15	4. Returning Output Files	How do I describe outputs from a command?
00:25	5. Capturing Standard Output	How do I capture a tool's standard output stream?

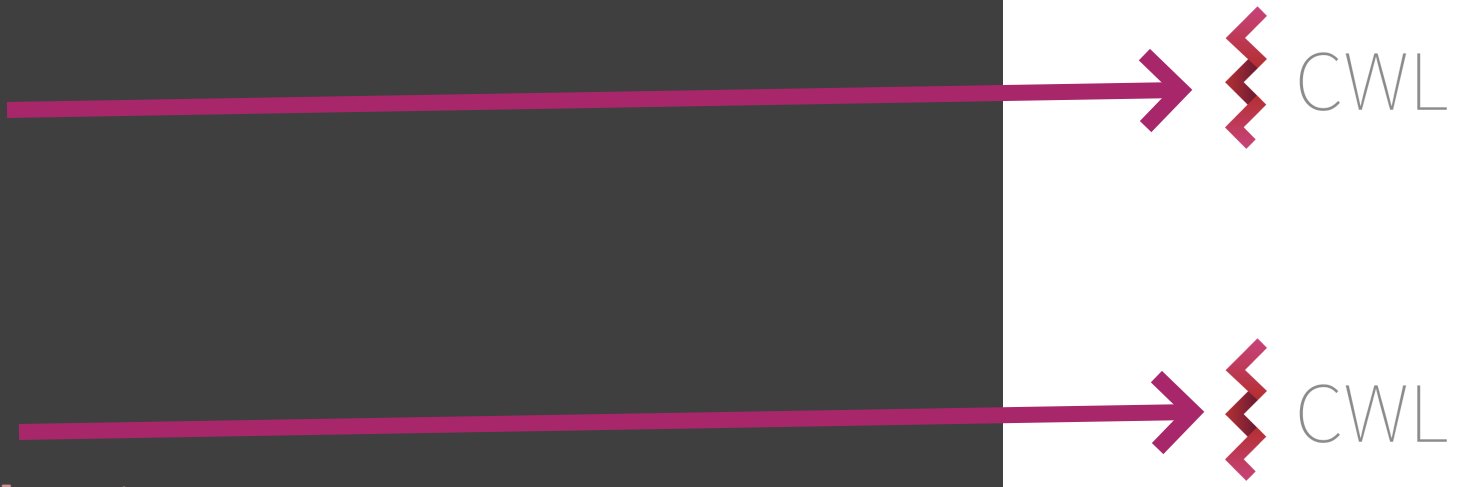


http://rabix.io/

SevenBridges

Composing a workflow

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow
inputs:
  inp: File
  ex: string

outputs:
  classout:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: inp
      extractfile: ex
    out: [example_out]

  compile:
    run: arguments.cwl
    in:
      src: untar/example_out
    out: [classfile]
```



https://www.ebi.ac.uk/metagenomics/ https://github.com/EBI-Metagenomics/ebi-metagenomics-cwl

```yaml
cwlVersion: v1.0
class: Workflow
label: EMG QC workflow, (paired end version). Benchmarking with MG-RAST expt.

requirements:
 - class: SubworkflowFeatureRequirement
 - class: SchemaDefRequirement
   types:
    - $import: ../tools/FragGeneScan-model.yaml
    - $import: ../tools/trimmomatic-sliding_window.yaml
    - $import: ../tools/trimmomatic-end_mode.yaml
    - $import: ../tools/trimmomatic-phred.yaml

inputs:
  reads:
    type: File
    format: edam:format_1930  # FASTQ

outputs:
  processed_sequences:
    type: File
    outputSource: clean_fasta_headers/sequences_with_cleaned_headers

steps:
  trim_quality_control:
```

EMBL-EBI

elixir

EBI-Metagenomics / **ebi-metagenomics-cwl**

Code | Issues 10 | Pull requests 1 | Projects 0 | Wik

Branch: **master** **ebi-metagenomics-cwl** / **workflows** /

mr-c more EDAM

..	
16S_taxonomic_analysis.cwl	add krona bits
cmsearch-multimodel.cwl	correct overlap clan name
convert-to-v3-layout.cwl	Update status of Toil & Directory type
ebi-setup.sh	add 1st party code to this repo
emg-assembly-job.yaml	improve metaspades resource reqs
emg-assembly.cwl	a few more renames
emg-core-analysis-v4.cwl	Removed trimmomatic from core ana
emg-pipeline-v3-paired-job.yaml	redo SSU matching steps
emg-pipeline-v3-paired.cwl	add krona bits
emg-pipeline-v3.cwl	more EDAM
emg-pipeline-v4-assembly-metaSPAd…	remove redundant input
emg-pipeline-v4-paired-job.yaml	Adding in additional steps and validat
emg-pipeline-v4-paired.cwl	remove redundant output

```
stain@biggie:~/src/ebi-metagenomics-cwl$ find . -name *cwl | xargs ls
./tools/5S-from-tablehits.cwl                ./tools/minia.cwl
./tools/biom-convert.cwl                     ./tools/modify_taxonomy_table.cwl
./tools/biom-summarize_table.cwl             ./tools/nhmmer.cwl
./tools/clean_fasta_headers.cwl              ./tools/oneLineFasta.cwl
./tools/cmsearch-deoverlap.cwl               ./tools/orf_stats.cwl
./tools/collate_unique_SSU_headers.cwl       ./tools/prepend_header.cwl
./tools/concatenate.cwl                      ./tools/pull-5Ss.cwl
./tools/count_fasta.cwl                      ./tools/pull-LSUs.cwl
./tools/count_fastq.cwl                      ./tools/pull-SSUs.cwl
./tools/create_categorisations.cwl           ./tools/qc-stats.cwl
./tools/discard_short_seqs.cwl               ./tools/qiime-filter_tree.cwl
./tools/esl-reformat.cwl                     ./tools/qiime-pick_closed_reference_
./tools/esl-sfetch-index.cwl                 ./tools/RevReadSort.cwl
./tools/esl-sfetch-manyseqs.cwl              ./tools/rRNA_selection.cwl
./tools/esl-sfetch-oneseq.cwl                ./tools/seqprep.cwl
./tools/extract_coord_lines.cwl              ./tools/seqprep-merge.cwl
./tools/extract-coords-from-cmsearch.cwl     ./tools/SSU-from-tablehits.cwl
./tools/extract_observations.cwl             ./tools/summary.cwl
./tools/extract_sig_coords.cwl               ./tools/trimmomatic.cwl
./tools/faselector.cwl                       ./tools/tRNA_selection.cwl
./tools/fasta_chunker.cwl                    ./tools/update_krona_chart_urls.cwl
./tools/fastq_to_fasta.cwl                   ./tools/write_ipr_summary.cwl
./tools/FragGeneScan1_20.cwl                 ./workflows/16S_taxonomic_analysis.c
./tools/go_summary.cwl                       ./workflows/cmsearch-multimodel.cwl
./tools/hmmsearch.cwl                        ./workflows/convert-to-v3-layout.cwl
./tools/infernal-cmscan.cwl                  ./workflows/emg-assembly.cwl
./tools/infernal-cmsearch.cwl                ./workflows/emg-core-analysis-v4.cwl
./tools/InterProScan5.21-60.cwl              ./workflows/emg-pipeline-v3.cwl
./tools/ipr_stats.cwl                        ./workflows/emg-pipeline-v3-paired.c
./tools/krona.cwl                            ./workflows/emg-pipeline-v4-assembly
./tools/krona_setup.cwl                      ./workflows/emg-pipeline-v4-paired.c
./tools/LSU-from-tablehits.cwl               ./workflows/emg-pipeline-v4-single.c
./tools/map_fa_headers.cwl                   ./workflows/emg-qc-paired.cwl
./tools/mapseq2biom.cwl                      ./workflows/emg-qc-single.cwl
./tools/mapseq.cwl                           ./workflows/functional_analysis.cwl
./tools/mask_RNA.cwl                         ./workflows/orf_prediction.cwl
./tools/megahit.cwl                          ./workflows/rna-selector.cwl
./tools/metaspades.cwl                       ./workflows/trim_and_reformat_reads.
```

Common Workflow Language Viewer

This tool visualises and lists the details of a CWL workflow with its inputs, outputs and steps and packages the files involved into a downloadable Research Object Bundle (zip file with metadata in a manifest), allowing it to be easily viewed and shared.

Want to make your workflows look their best in CWL Viewer? Find out about CWL recommended practices

Workflow URL

Provide a Github, Gitlab or Git repository link to the workflow (or directory of workflows) here

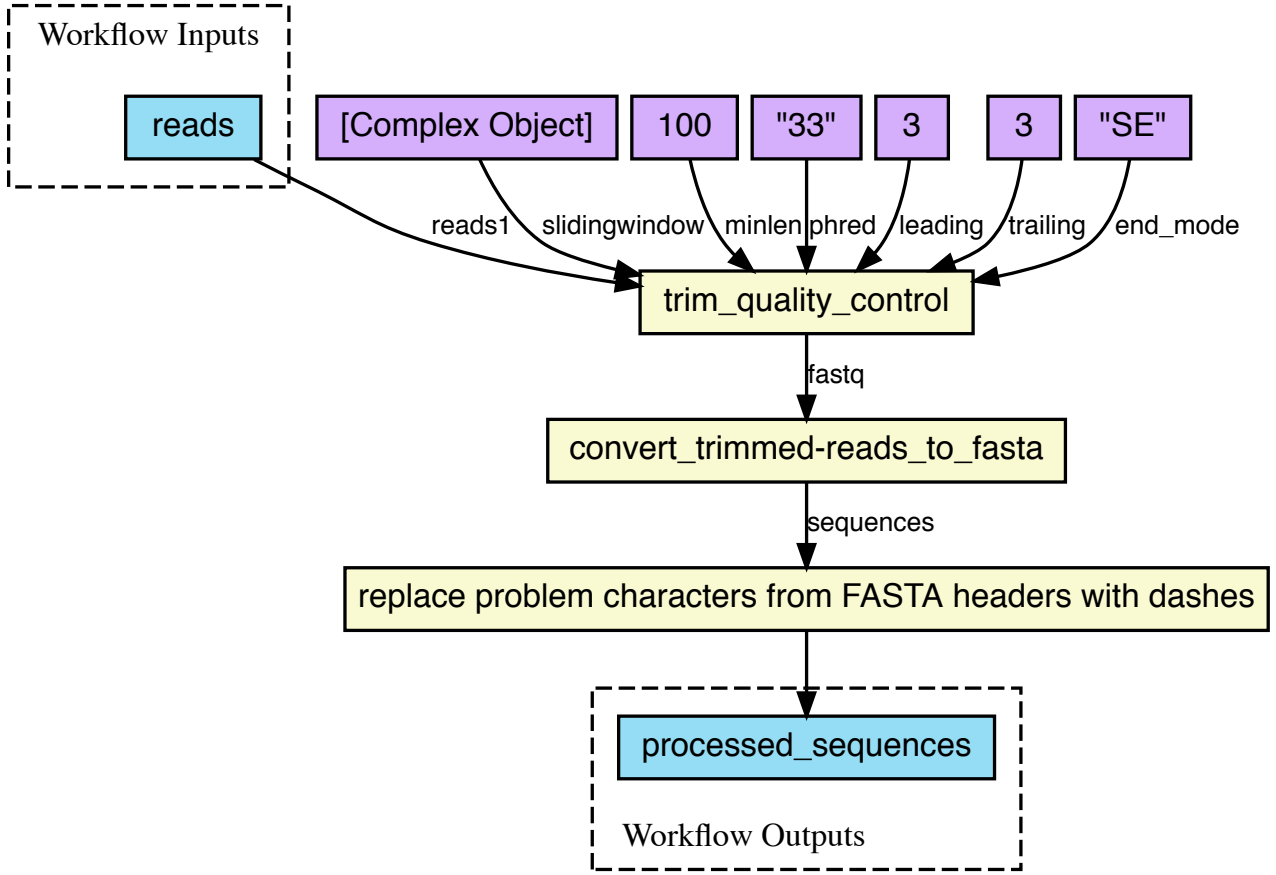
Don't know what to view? Try these from *common-workflow-language/workflows*: compile, make-to-cwl, lobSTR or explore the collection

URL to workflow

https://github.com/EBI-Metagenomics/ebi-metagenomics-cwl/blob/master/workflows/emg-pipeline-v4-single.cwl

Parse Workflow

https://view.commonwl.org/



https://w3id.org/cwl/view/git/886df9de6713e06228d2560c40f451155a196383/workflows/emg-qc-single.cwl

Provenance for Workflows?

Prospective provenance

The ‘recipes’ used to execute a computational task, e.g. the workflow **specification** or *workflow template.*

Retrospective provenance

The **execution record** of the running a workflow instance, details of every executed process and comprehensive info on execution environment

Workflow evolution

Changes across the workflow *specification*, its *tool* definitions and the *software* and *services* it depends on.

common-workflow-language / **workflows**

Code | Issues 13 | Pull requests 11 | Projects 0 | Wiki | Insights

Tree: 933bf2a1a1 **workflows** / workflows / **hello** /

tetron Update workflows/hello to v1.0

..

README.md	hello with param
hello-param.cwl	Update workflows/hello to v1.0
hello.cwl	Update workflows/hello to v1.0
params.js	update to draft3 final

README.m

Workflow URL

https://view.commonwl.org

Provide a Github, Gitlab or Git repository link to the workflow (or directory of workflows) here

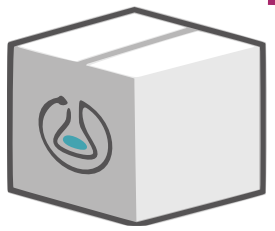
Don't know what to view? Try these from *common-workflow-language/workflows*: compile, make-to-cwl, lobSTR or explore the collection

URL to workflow

https://github.com/common-workflow-language/workflows/blob/master/workflow

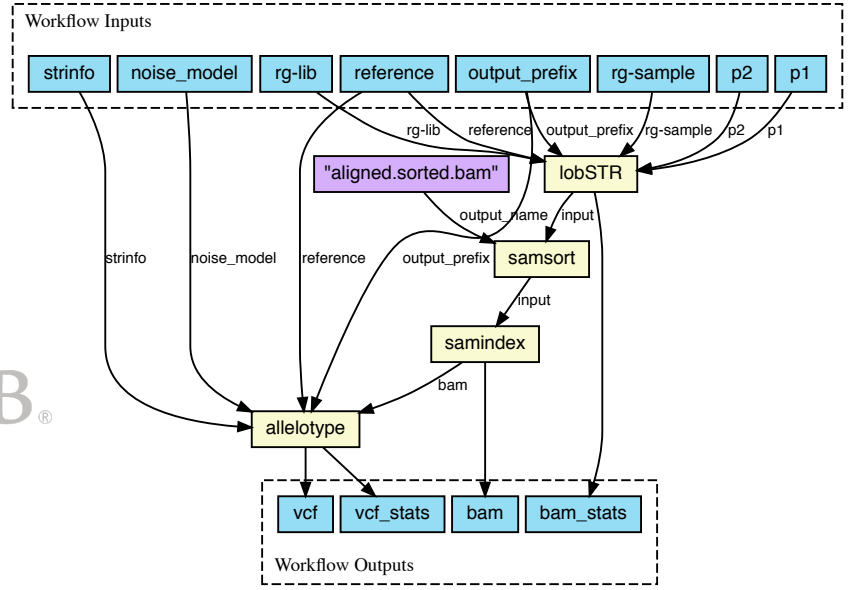
Parse Workflow

CWL



mongoDB.

Apache Jena

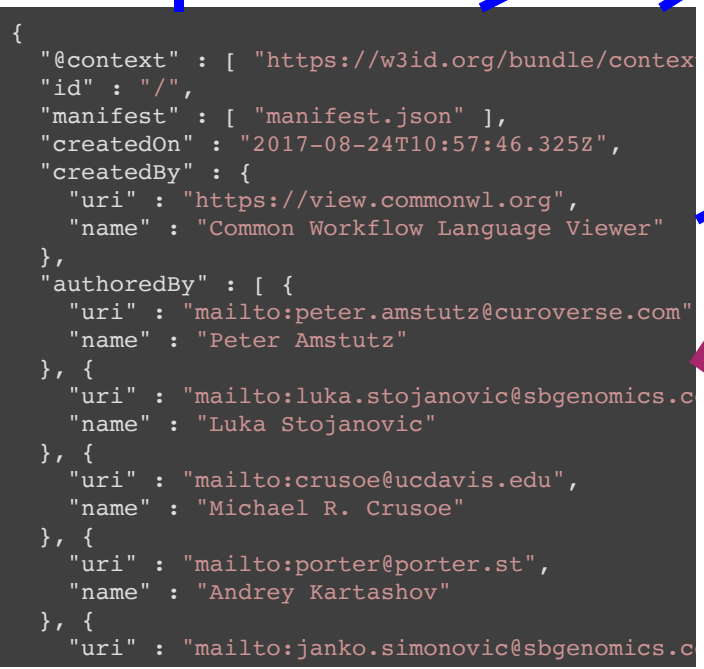


https://w3id.org/cwl/view/git/933.../workflows/hello/hello.cwl#main

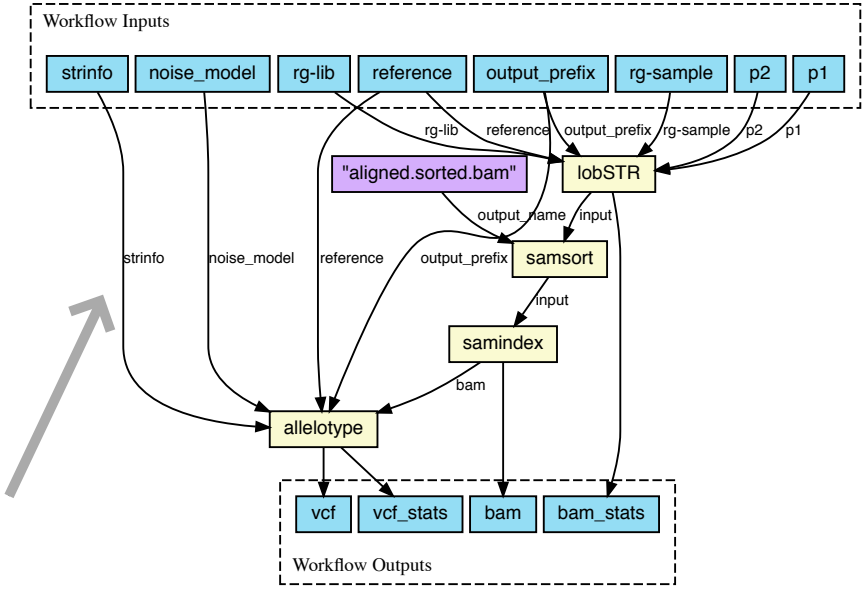
```
#!/usr/bin/env cwl-runner
cwlVersion: v1.0
class: Workflow

label: "Hello World"
doc: "Outputs a message using echo"

inputs: []

outputs:
  response:
    outputSource: step0/response
    type: File

steps:
  step0:
    run:
      class: CommandLineTool
      inputs:
        message:
          type: string
          doc: "The message to print"
          default: "Hello World"
```

CWL

https://view.commonwl.org/



ORCID



https://w3id.org/cwl/view/git/933.../workflows/hello/hello.cwl#main

https://w3id.org/cwl/view

Permalink URI scheme

`https://w3id.org/cwl/view/{scheme}/{commit}/{path}{?part=fragment}`

- *https://w3id.org/cwl/view/* fixed prefix at permalink service https://w3id.org/
- {scheme} - source code management protocol, currently only **git** supported:
  - *{commit}* - full git **commit** sha1 id (no branches or short commits allowed)
  - *{path}* - relative path to .cwl file within a checkout of that git commit
  - *{?part=fragment}* - optional part within CWL file , e.g. *#main*

Any git permalinks are resolved using https://view.commonwl.org/git which - if it knows about that particular git commit - will **content-negotiate** to provide various representations.

Anyone can mint these permalinks for .cwl files for a *given commit*, in any public or private git repository, given no uncommitted files or git submodules.

https://w3id.org/cwl/view/git/886df9de6713e06228d2560c40f451155a196383/workflows/emg-qc-single.cwl

CWLProv

Retrospective provenance capture and its challenges

https://www.commonwl.org/
common-workflow-language/cwltool

Farah Zaib Khan, Stian Soiland-Reyes, Richard O. Sinnott, Andrew Lonie, Michael R. Crusoe

The University of Melbourne, Australia
The University of Manchester, UK
Common Workflow Language Project

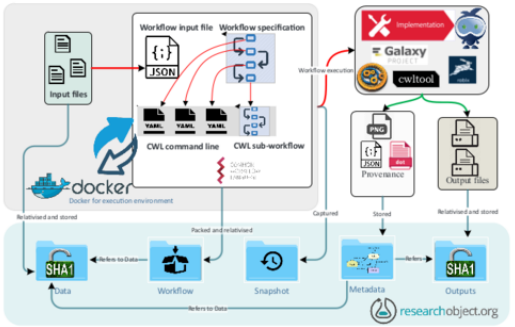
Capturing workflow provenance

CWLProv adds provenance capture to **cwltool**, the reference implementation of CWL.

Snapshots of the workflow and input files are captured in the **Research Object** (RO) with relativizing paths. The generated **input JSON file** allows re-execution.

Data generated by the workflow run, including intermediates, are SHA1-hashed and added to the RO with **content-addressable** identifiers.

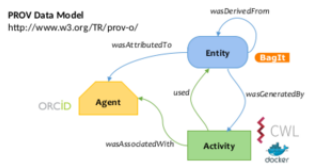
The retrospective **provenance trace** of the run refers to data using their hash, linking to the workflow definition.



Workflow Provenance Profile

L	PROV type	Subtype	Relations	
1	**Plan**	wfdesc:Workflow	wfdesc:hasSubProcess	wfdesc:Process
1		wfdesc:Process		
0	**Activity**	wfprov:WorkflowRun	wasAssociatedWith	wfprov:WorkflowEngine
1			hadPlan	wfdesc:Workflow
0			wasStartedBy	wfprov:WorkflowEngine
0			atTime	(ISO8601 time:stamp)
2			wasStartedBy	wfprov:WorkflowRun
0			wasEndedBy	wfprov:WorkflowEngine
0			atTime	(ISO8601 time:stamp)
1		wfprov:ProcessRun	wasStartedBy	wfprov:WorkflowRun
1			atTime	(ISO8601 time:stamp)
1			used	wfprov:Artifact
1			role	wfdesc:InputParameter
1			wasAssociatedWith	wfprov:WorkflowRun
1			hadPlan	wfdesc:Process
1			wasEndedBy	wfprov:WorkflowRun
1			atTime	(ISO8601 time:stamp)
2		SoftwareAgent	wasAssociatedWith	wfprov:ProcessRun
2			cwlprov:image	(docker image)
0	**SoftwareAgent**	wfprov:WorkflowEngine	wasStartedBy	Person (ORCID)
0			label	(cwltool -version)
1	**Entity**	wfprov:Artifact	wasGeneratedBy	wfprov:ProcessRun
1			role	wfdesc:OutputParameter

CWLProv profile of **W3C PROV**, *extended with Research Object Model* **wfdesc** *(prospective provenance) and* **wfprov** *(retrospective provenance). Indentation indicates n-ary relationships (hadPlan/atTime/role/image).*



Best Practices for Publishing Workflow Analyses

R1: Workflows should be treated as **first class data objects** (Corcho 2012).

R2: Workflow specification alone is insufficient to ensure reusability of scientific experiments (Zhao 2012). Complete provenance **capture of workflow enactment** should be published along with the workflow specification (Garijo 2017). This can also help to avoid *workflow decay*.

R3: A **structured description** of the experimental steps carried out in a workflow using a "system-neutral" language can ensure well-documented and well described workflows for enhanced understandability of methods (Belhajjame 2015).

R4: Availability of the **underlying software** needed by each step of a given workflow is critical. (Kanwal 2017). More recently **container** technologies such as *Docker*, *OpenVZ* or *LXC* can be exploited to package the environment and configuration together.

R5: While publishing digital scholarly objects, **open licensing** should be adapted as a practice to allow *sharing* and *reproducing* of published analyses (Stodden 2016).

R6: The description of the underlying software is not enough for reproducing an analysis; instead **workflow specifications** and **configuration** should also be published (Garijo 2017).

R7: **Intermediate data** products should be captured (if feasible) to facilitate debugging, error handling and thorough examination of the published workflows and associated results (Sandve 2013).

https://f1000research.com/posters/7-916

Common Workflow Language

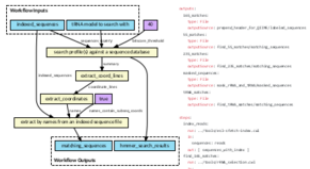
Common Workflow Language (CWL) aims to provide extensible, open source standards supporting interoperable, portable and reproducible workflow-based research (Amstutz 2016).

CWL is a community-driven effort, being adapted by leading workflow design and execution platforms including UCSC' *Toil*, Curoverse's *Arvados*, Seven Bridges' *Rabix*, Broad Institute's *Cromwell*, IBM's *CWLExec*, *Galaxy* and *Apache Taverna* (incubating).

CWL provides declarative constructs for workflow and command line tool definition and make minimal assumptions about base software dependencies, configuration settings, software versions, parameter settings or the execution environment more generally.

To a large extent CWL achieves workflow **portability** by executing bioinformatics tools distributed as **Docker** containers or **BioConda** packages, ensuring the correct tool version and its dependencies are installed. Defining tool descriptions in CWL encourages reuse of steps across workflows and researchers.

The CWL community commonly develops workflows as open source in GitHub repositories, which can be visualized in the **CWL Viewer**. CWL definitions can be repurposed or reused across multiple labs, as owing to the **interoperability** aspect of CWL they do not need to agree on workflow engine or compute architecture.



tRNA selection, EMG pipeline (Mitchell 2018)
Rob Finn, Michael Crusoe
EBI-Metagenomics/ebi-metagenomics-cwl

Provenance levels

The granularity level of the workflow execution in the RO can be classified into four levels to realize a hierarchical and modular solution towards interoperable domain-specific provenance.

Level 0 adds **snapshots** of the files used to start a workflow enactment and optionally the final outputs. This can be used for result interpretation (e.g. workflow parameters) and facilitates *re-executing* or *repurposing* the workflow as a whole. In the simplest case this logs the workflow engine command line arguments; in CWLProv we create a re-executable *cwl-runner* **job file** with relative links to inputs and workflow definition files.

Level 1 adds *retrospective provenance* with *content-addressable* data to enable reproducibility of the main analysis, step by step. In CWLProv this takes the form of a **W3C PROV** file. Activities link to local identifiers of steps within the workflow *plan*, providing *prospective provenance* (Zhao 2006).

Level 2 (work in progress) logs *within steps*, e.g. the exact Docker image ids fetched, as well as adding retrospective provenance of **nested workflows.** Although technically this can be added to the same PROV file as level 1, we found it cleaner with separate PROV files per step execution so each step is re-runnable independently.

Level 3 (planned) adds *augmented* and *filtered* provenance, **domain-specific** and focused only on data/steps relevant to the researcher. In CWLProv we will build on existing work like *CWL Viewer*, *LabelFlow* (Alper 2018) and *BioCompute Objects* (BCO) to extract CWL annotations using domain-specific ontologies (**EDAM**, **BioSchemas**) apply them to the produced data, and classify steps into *motifs* like data retrieval, preparation, analyses and visualization (Garijo 2014).

CWLProv Research Objects

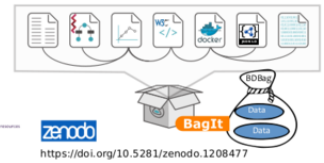
One challenge when capturing or distributing details of a workflow enactment is handling *multiple files* (workflow, scripts, datasets), *non-digital* resources (e.g. people, equipment, samples), execution *dependencies* (Bechhofer 2013) as **Docker** or **BioConda** packages.

These resources often have separate *identifiers*, *versions*, *attributions* and can be produced by humans or automated processes. Large data (e.g. metagenomic reads) can be non-trivial to transfer correctly or efficiently, and may reside in multiple repositories (Madduri 2018).

Research Objects encapsulate the digital artefacts associated with a given computational analysis, richly described with structured metadata in a **JSON-LD manifest**. Aggregated resources may include *input* and *output* data for experiment results; computational *methods* such as command line tools and workflow specifications; *attribution* details; retrospective and prospective *provenance*; and machine-readable *annotations* regarding the included artefacts and the relation between them (Belhajjame 2015).

The RO approach, developed in full for **workflow preservation** by capturing provenance of executing bioinformatics (Hettne 2014) and virtual astronomy workflows (Ruiz 2014), has expanded into other domains to cover investigations and datasets in systems biology (Wolstencroft 2013), earth sciences (Garcia-Silva 2017), health informatics (Pavis 2015) and precision medicine (Alterovitz 2018).

CWLProv updates the original *Workflow Research Object* profile with the **BagIt** serialization (Kunze 2018), a Library of Congress supported digital archive mechanism that emphasizes file authenticity and completeness. RO and BagIt, combined with the **MinId** distributed identification resolution mechanism, forms **BDBag** (Big Data Bag), that can capture and transfer datasets from large-scale genomics workflow runs (Chard 2016), used in *NIH Data Commons*.



https://doi.org/10.5281/zenodo.1208477

We gratefully acknowledge **The University of Melbourne** for providing funding support as MIRS and MIFRS scholarship.

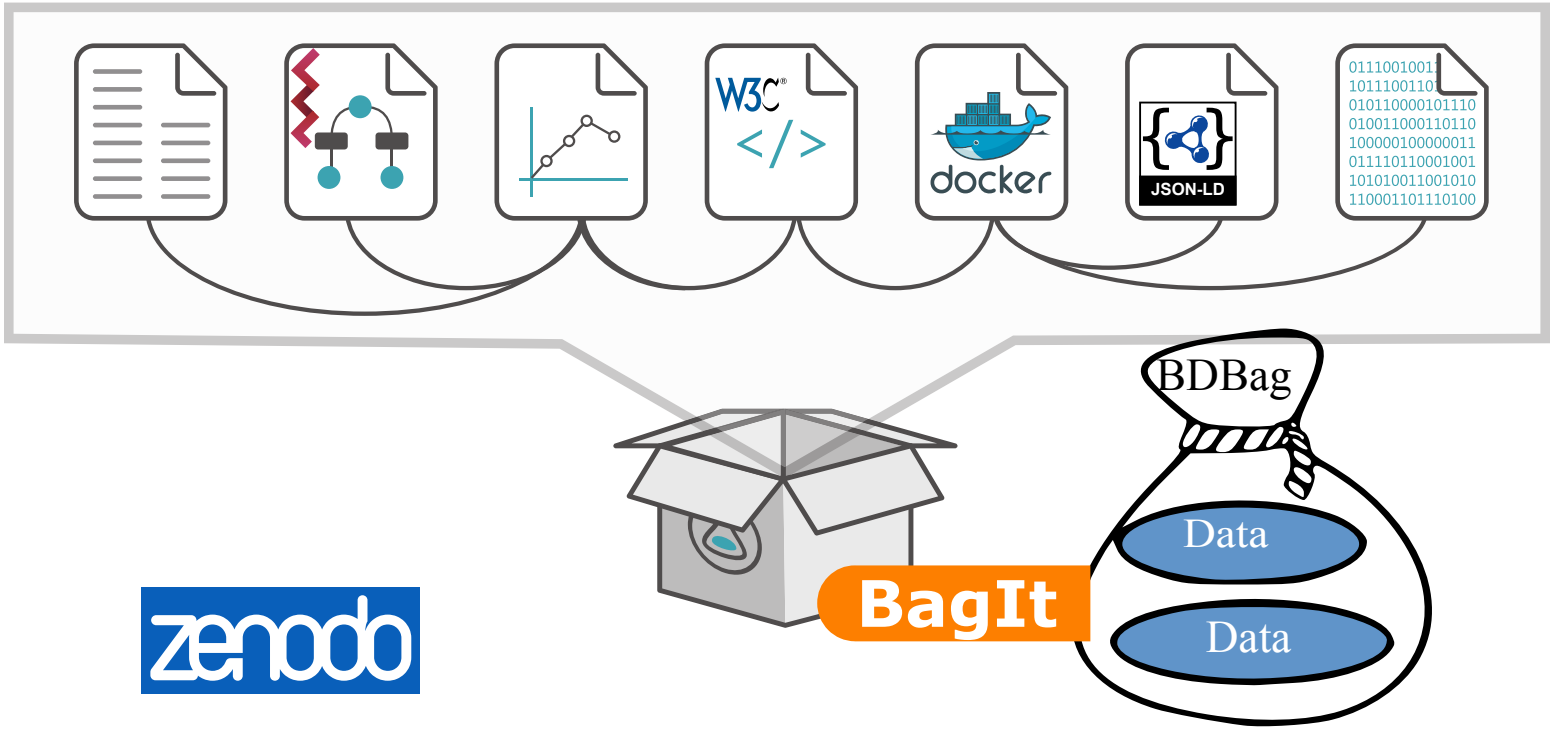
This work has been done as part of the **BioExcel CoE** (www.bioexcel.eu), a project funded by the European Union contract H2020-EINFRA-2015-1-675728.

Farah Zaib Khan et al,

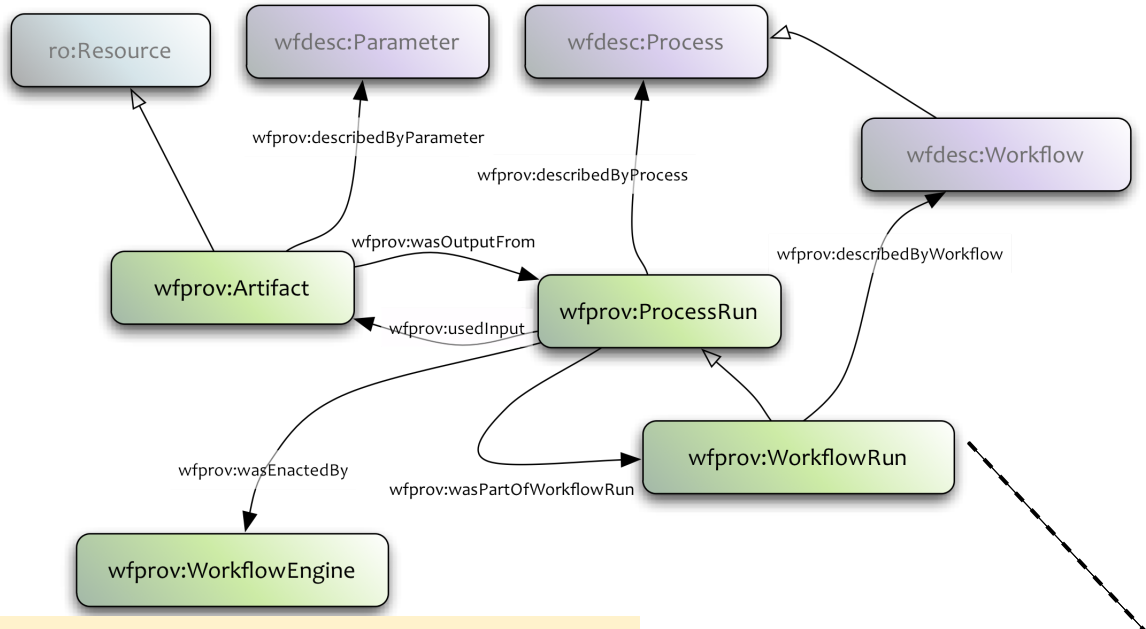
CWLProv – Interoperable retrospective provenance capture and its challenges,

BOSC 2018

https://doi.org/10.7490/f1000research.1115721.1



https://doi.org/10.5281/zenodo.1208477



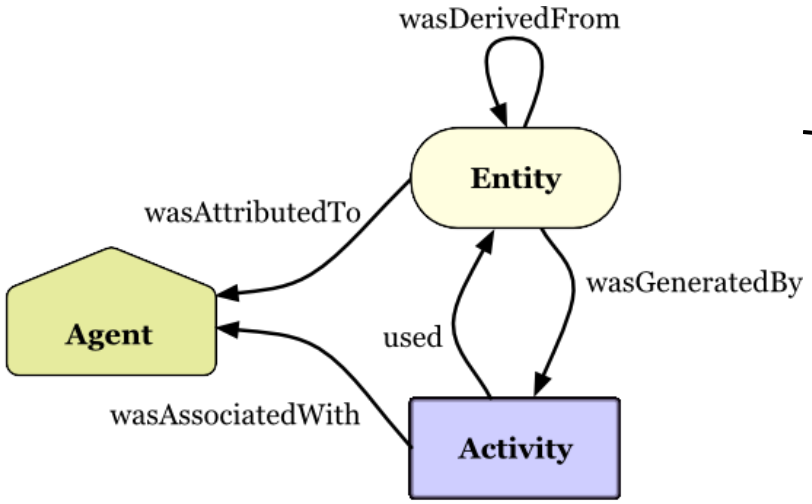
Workflow specifications

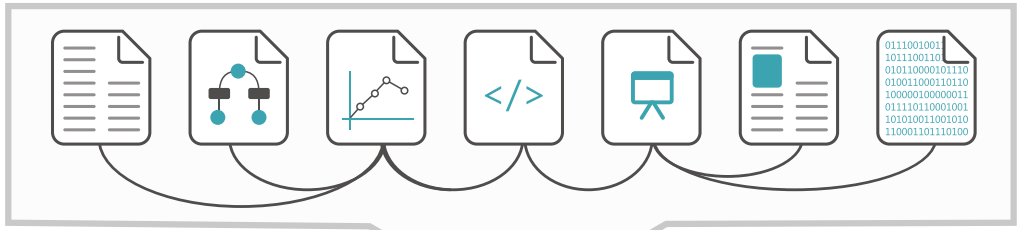
COMMON WORKFLOW LANGUAGE

docker

Provenance using PROV-Model

expanded with wfprov and wfdesc



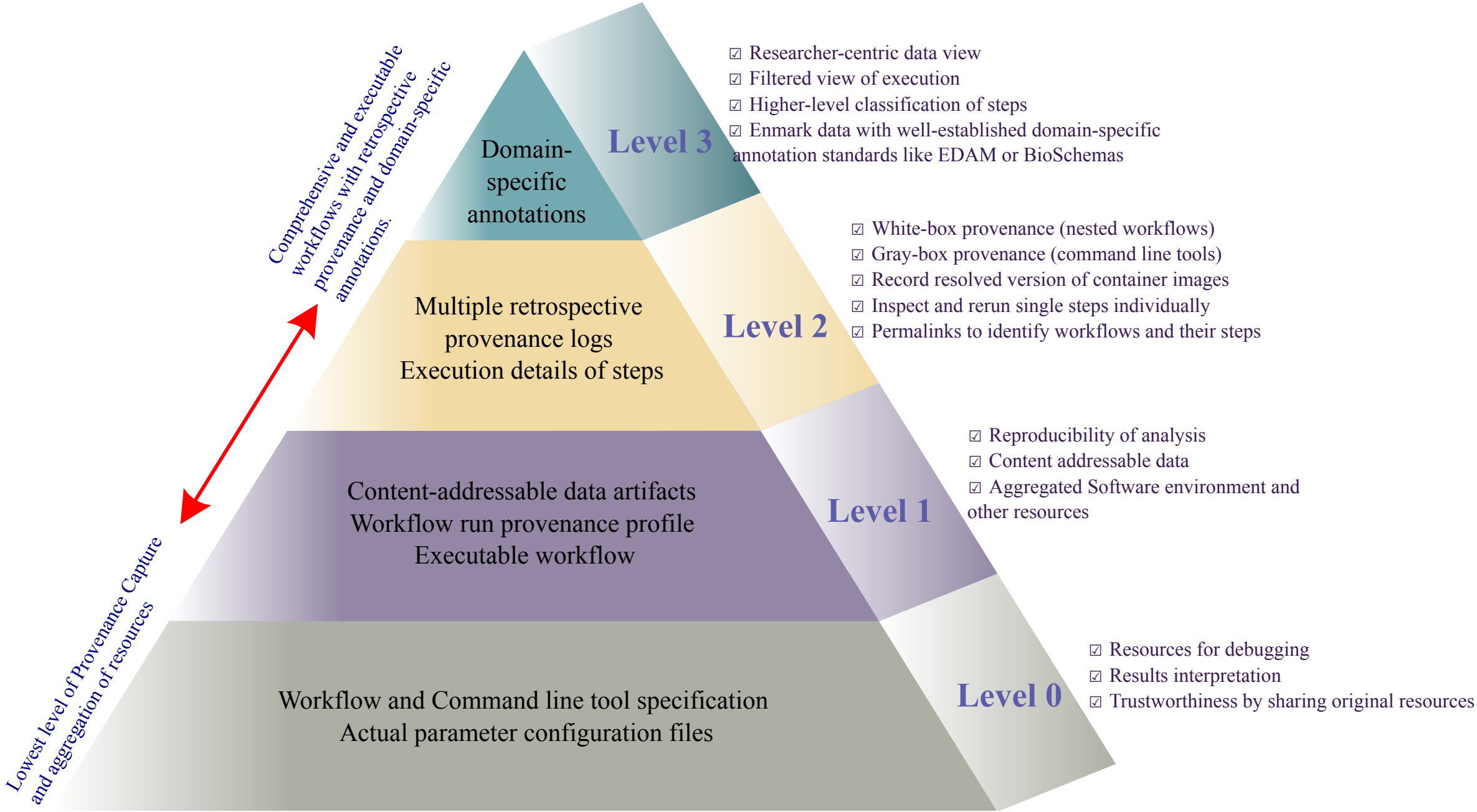


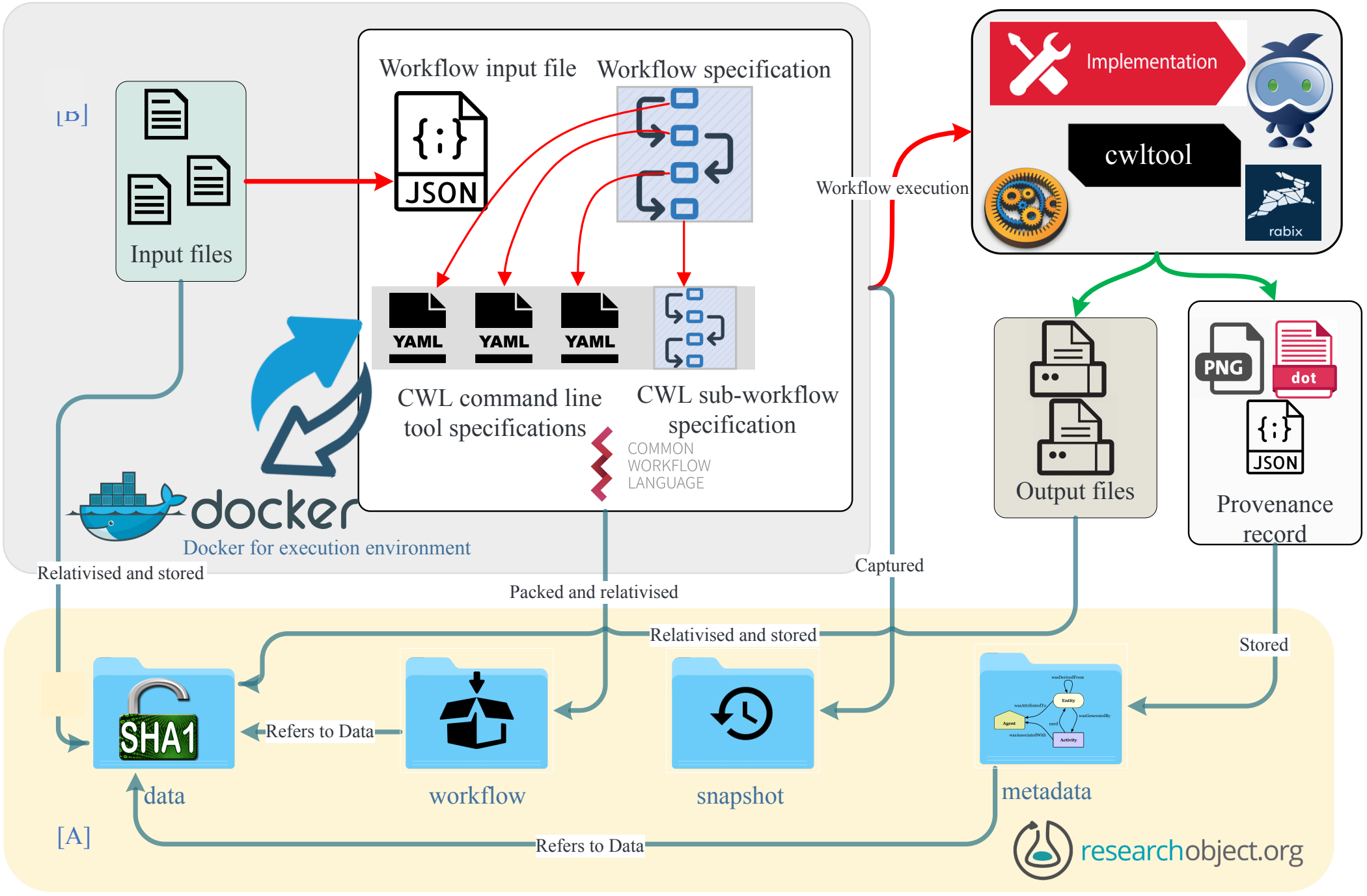
researchobject.org

Enabling reproducible, transparent research

https://doi.org/10.5281/zenodo.1473157

Levels of Provenance




[B]
Input files
Workflow input file
Workflow specification
JSON
YAML
YAML
YAML
CWL command line tool specifications
CWL sub-workflow specification
COMMON WORKFLOW LANGUAGE
docker
Docker for execution environment
Workflow execution
Implementation
cwltool
rabix
Output files
PNG
dot
JSON
Provenance record
Relativised and stored
Packed and relativised
Captured
Relativised and stored
Stored
SHA1
Refers to Data
data
workflow
snapshot
metadata
Refers to Data
[A]
researchobject.org

```
./revsort-run-1/bagit.txt
./revsort-run-1/bag-info.txt

./revsort-run-1/manifest-sha1.txt
./revsort-run-1/tagmanifest-sha1.txt

./revsort-run-1/data
./revsort-run-1/data/97/97fe1b50b4582cebc7d853796ebd62e3e163aa3f
./revsort-run-1/data/32/327fc7aedf4f6b69a42a7c8b808dc5a7aff61376
./revsort-run-1/data/b9/b9214658cc453331b62c2282b772a5c063dbd284

./revsort-run-1/workflow
./revsort-run-1/workflow/packed.cwl
./revsort-run-1/workflow/primary-output.json
./revsort-run-1/workflow/primary-job.json

./revsort-run-1/snapshot
./revsort-run-1/snapshot/revtool.cwl
./revsort-run-1/snapshot/revsort.cwl
./revsort-run-1/snapshot/sorttool.cwl

./revsort-run-1/metadata/logs/engine.ac9c1653-4291-47bc-86f8-6dedcff13519.txt

./revsort-run-1/metadata/manifest.jsonld

./revsort-run-1/metadata/provenance/primary.cwlprov.provn
./revsort-run-1/metadata/provenance/primary.cwlprov.json
./revsort-run-1/metadata/provenance/primary.cwlprov.ttl
./revsort-run-1/metadata/provenance/primary.cwlprov.jsonld
./revsort-run-1/metadata/provenance/primary.cwlprov.xml
./revsort-run-1/metadata/provenance/primary.cwlprov.nt
```

Profile

Completeness

Workflow values

Rerunnable

Reusable

Unstructured log

Relations and identifiers

Structured execution log

"Transport level"

Level 0

Level 1

PROV Data Model

http://www.w3.org/TR/prov-o/

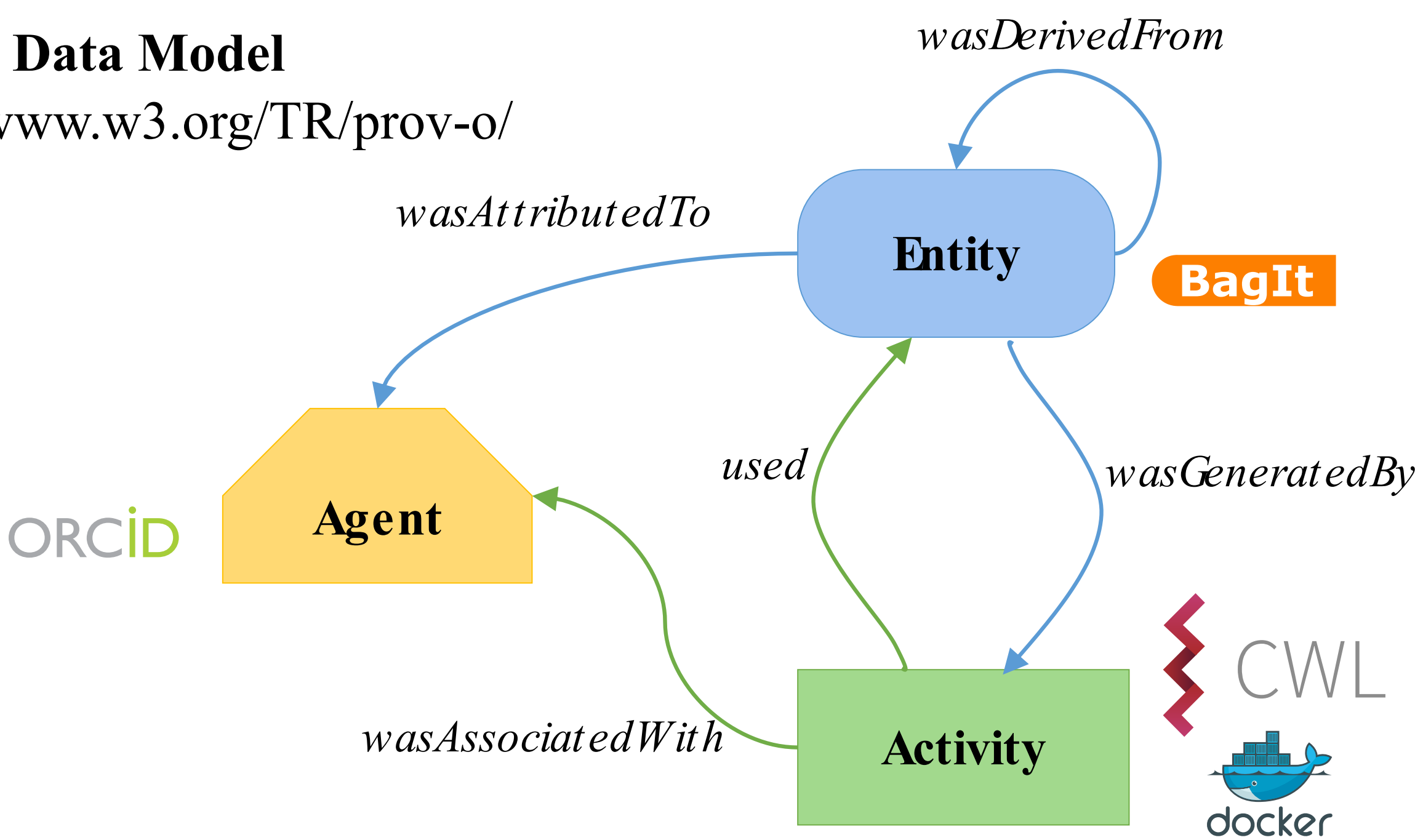


Table 2. *CWLProv* profile of W3C PROV, extended with Research Object Model's *wfdesc* (prospective provenance) and *wfprov* (retrospective provenance).

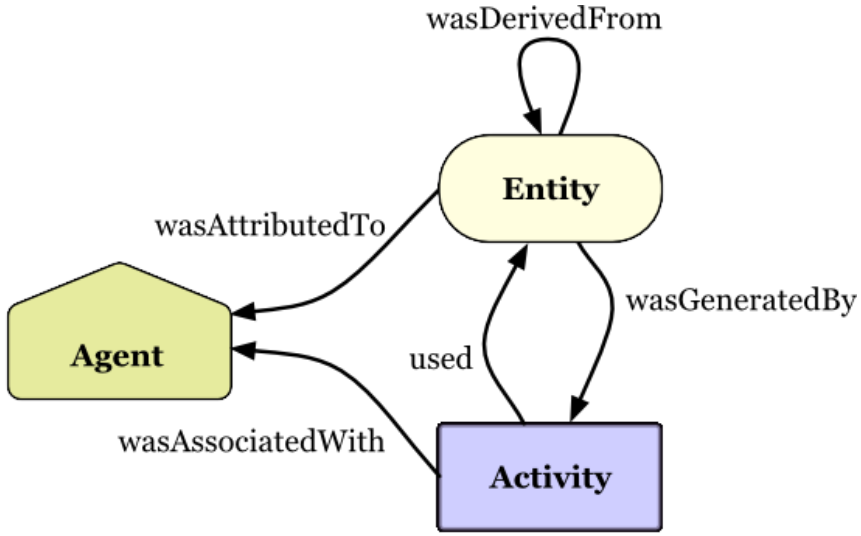
https://doi.org/10.5281/zenodo.1473157

Level	PROV type	Subtype	Relation	Range
1	**Plan**	wfdesc:Workflow	wfdesc:hasSubProcess	wfdesc:Process
1		wfdesc:Process		
0	**Activity**	wfprov:WorkflowRun	wasAssociatedWith	wfprov:WorkflowEngine
1			↳ hadPlan	wfdesc:Workflow
0			wasStartedBy	wfprov:WorkflowEngine
0			↳ atTime	*ISO8601 timestamp*
2			wasStartedBy	wfprov:WorkflowRun
1			wasEndedBy	wfprov:WorkflowEngine
0			↳ atTime	*ISO8601 timestamp*
1		wfprov:ProcessRun	wasStartedBy	wfprov:WorkflowRun
1			↳ atTime	*ISO8601 timestamp*
1			used	wfprov:Artifact
1			↳ role	wfdesc:InputParameter
1			wasAssociatedWith	wfprov:WorkflowRun
1			↳ hadPlan	wfdesc:Process
1			wasEndedBy	wfprov:WorkflowRun
1			↳ atTime	*ISO8601 timestamp*
2		SoftwareAgent	wasAssociatedWith	wfprov:ProcessRun
2			↳ *CWLProv*:image	*docker image id*
0	**SoftwareAgent**	wfprov:WorkFlowEngine	wasStartedBy	Person *ORCID*
0			label	*cwltool --version*
1	**Entity**	wfprov:Artefact	wasGeneratedBy	wfprov:Processrun
1			↳ role	wfdesc:OutputParameter
1	**Collection**	wfprov:Artefact	hadMember	wfprov:Artefact
1		Dictionary	hadDictionaryMember	wfprov:Artefact
1			↳ pairKey	*filename*

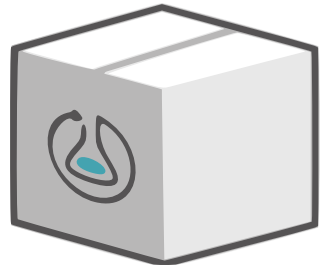
Indentation with ↳ indicates n-ary relationships which are expressed differently depending on PROV syntax. Namespaces: `http://www.w3.org/ns/prov#` (default), `http://purl.org/wf4ever/wfdesc#` (*wfdesc*), `http://purl.org/wf4ever/wfprov#` (*wfprov*), `https://w3id.org/cwl/prov#` *CWLProv*)

```
document
prefix wfprov <http://purl.org/wf4ever/wfprov#>
prefix prov <http://www.w3.org/ns/prov#>
prefix wfdesc <http://purl.org/wf4ever/wfdesc#>
prefix wf <https://w3id.org/cwl/view/git/933bf2a1a1cce32d88f88f136275535da9df0954/wo
prefix input <app://579c1b74-b328-4da6-80a8-a2ffef2ac9b5/workflow/input.json#>
prefix run <urn:uuid:>
prefix engine <urn:uuid:>
prefix data <urn:hash:sha256:>

default <app://579c1b74-b328-4da6-80a8-a2ffef2ac9b5/>

// Level 1 provenance of workflow run

activity(run:2e1287e0-6dfb-11e7-8acf-0242ac110002, , , [prov:type='wfprov:WorkflowRu
    wasStartedBy(run:2e1287e0-6dfb-11e7-8acf-0242ac110002, -, -, -, 2017-10-27T14:24

    // The engine is the SoftwareAgent that is executing our Workflow plan
    wasAssociatedWith(run:2e1287e0-6dfb-11e7-8acf-0242ac110002, engine:b2210211-8acb
        agent(engine:b2210211-8acb-4d58-bd28-2a36b18d3b4f, prov:type='prov:SoftwareA
        // prov has no term to relate sub-plans - we'll use wfdesc:hasSubProcess
        entity(wf:main,[prov:type='wfdesc:Workflow', prov:type='prov:Plan', wfdesc:h
            alternateOf(wf:main, workflow/packed.cwl)
            entity(wf:main/step1,[prov:type='wfdesc:Process', prov:type='prov:Plan']
            entity(wf:main/step2,[prov:type='wfdesc:Process', prov:type='prov:Plan']

    // First the workflow uses some data; here with a urn:sha:sha256 identifier
    used(run:2e1287e0-6dfb-11e7-8acf-0242ac110002, data:5891b5b522d5df086d0ff0b110fb
        entity(data:5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
            // which we have stored a copy of within the research object
            specializationOf(data/58/5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d

    // Then there was another activity - wfprov:ProcessRun indicating a command line
    activity(run:4305467e-6dfb-11e7-885d-0242ac110002, -, -, [prov:type='wfprov:Proc
```

CWLProv



BagIt



https://doi.org/10.5281/zenodo.1215611

Which PROV format?

PROV-N

```
wasGeneratedBy(ent1, act1,
  2017-10-26T21:32:52Z, ex:port="p1")
```

PROV-XML

```
<prov:wasGeneratedBy>
  <prov:entity prov:ref="ex:ent1"/>
  <prov:activity prov:ref="ex:act1"/>
  <prov:time>2017-10-26T21:32:52Z</prov:time>
  <ex:port>p1</ex:port>
</prov:wasGeneratedBy>
```

PROV-JSON

```
    "wasGeneratedBy": {
        "ex:gen1": {
            "prov:entity": "ent1",
            "prov:activity": "act1",
            "prov:time": "2017-10-26T21:32:52Z",
            "ex:port": "p1"
        },
    },
```

PROV-O Turtle

```
:ent1
  a prov:Entity;
  prov:wasGeneratedBy :act1;
  prov:generatedAtTime "2017-10-26T21:32:52Z"^^xsd:dateTime ;
  ex:port "p1" .
```

PROV-O JSON-LD

```
{ "@context": { .. },
  "@id": "ent1",
  "@type": "prov:Entity",
  "ex:port": "p1",
  "prov:generatedAtTime":  "2017-10-26T21:32:52Z",
  "prov:wasGeneratedBy": {
    "@id": "act1",
    "@type": "prov:Activity"
  }
}
```

https://www.w3.org/TR/prov-overview/

Nested workflows

```
prefix id <urn:uuid:>
prefix provenance <arcp://uuid,73eab018-7b36-4f84-a845-aca8073bd46c/metadata/provenance/>

agent(id:a606d227-bf10-4479-8d11-823bb932bbac,
    [prov:type='wfprov:WorkflowEngine', prov:type='prov:SoftwareAgent',
     prov:label="cwltool 1.0.20180817162414"])

activity(id:73eab018-7b36-4f84-a845-aca8073bd46c, 2018-08-21T15:20:35.059920, -,
    [prov:type='wfprov:WorkflowRun', prov:label="Run of workflow/packed.cwl#main"])
wasStartedBy(id:73eab018-7b36-4f84-a845-aca8073bd46c, -, id:a606d227-bf10-4479-8d11-823bb932bbac, 2018-08-21T15:20:35.060038)

activity(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, -,
     [prov:type='wfprov:ProcessRun', prov:label="Run of workflow/packed.cwl#main/compile"])
wasStartedBy(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, id:73eab018-7b36-4f84-a845-aca8073bd46c, 2018-08-21T15:20:35.163189)

activity(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, -,
     [prov:has_provenance='provenance:workflow_compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.provn',
      prov:has_provenance='provenance:workflow_compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.ttl'
])
```

metadata/provenance/primary.cwlprov.provn

```
{
    "about": "urn:uuid:e79fc8dc-6e40-4236-b22c-41fee22947a9",
    "content": [
        "provenance/workflow_20compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.provn",
        "provenance/workflow_20compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.ttl",
    ],
    "oa:motivatedBy": {
        "@id": "http://www.w3.org/ns/prov#has_provenance"
    }
}
```

metadata/manifest.json

metadata/provenance/
primary.cwlprov.provn

```
activity(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, -,
     [prov:type='wfprov:ProcessRun', prov:label="Run of workflow/packed.cwl#main/compile"])
wasStartedBy(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, id:73eab018-7b36-4f84-a845-aca8073bd46c, 2018-08-

activity(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, -,
     [prov:has_provenance='provenance:workflow_compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.provn'
      prov:has_provenance='provenance:workflow_compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.ttl'
])
```

Level 1

metadata/provenance/
workflow_compile.e79fc8dc-6e40-4236-b22c-41fee22947a9.cwlprov.provn

```
prefix id <urn:uuid:>
agent(id:a606d227-bf10-4479-8d11-823bb932bbac,
    [prov:type='wfprov:WorkflowEngine', prov:type='prov:SoftwareAgent',
     prov:label="cwltool 1.0.20180817162414"])

activity(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, 2018-08-21T15:20:35.089187, -,
    [prov:type='wfprov:WorkflowRun', prov:label="Run of workflow/packed.cwl#compile.cwl"])
wasStartedBy(id:e79fc8dc-6e40-4236-b22c-41fee22947a9, -, id:a606d227-bf10-4479-8d11-823bb932bbac, 2018-08-

activity(id:9b1a2b69-3403-4063-9ff0-e7ac2df32036, -, -,
     [prov:type='wfprov:ProcessRun', prov:label="Run of workflow/packed.cwl#compile.cwl/step1"])
wasStartedBy(id:9b1a2b69-3403-4063-9ff0-e7ac2df32036, -, id:e79fc8dc-6e40-4236-b22c-41fee22947a9, 2018-08-

activity(id:9b1a2b69-3403-4063-9ff0-e7ac2df32036, -, -,
     [prov:type='wfprov:ProcessRun', prov:label="Run of workflow/packed.cwl#compile.cwl/step2"])
```

Level 2

Identifying intermediate data

Output 1B file is also Input 2C and Input 3D downstream

Simple filenames -> duplications

`./data/step1/outputB.txt`

`./data/step2/inputC.txt`

`./data/step3/inputD.txt`

Content-adressable

SHA-256 hash of bytes as **filename:**

`./data/51/51fb8af0c4ae0422fbe88340d91880ecb9d7537cf57339c1cf1256b7ca58f32d`

RFC6920 **URI** as global identifier:

`nih:sha-256;51fb8af0c4ae0422fbe88340d91880ecb9d7537cf57339c1cf1256b7ca58f32d`

metadata/manifest.json


```
{
    "@context": [
        {
            "@base": "arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/metadata/"
        },
        "https://w3id.org/bundle/context"
    ],

    "id": "/",
    "conformsTo": "https://w3id.org/cwl/prov/0.6.0",
    "manifest": "manifest.json",
    "createdOn": "2018-10-25T15:46:43.191346",
    "createdBy": {
        "uri": "urn:uuid:ac9c1653-4291-47bc-86f8-6dedcff13519",
        "name": "cwltool 1.0.20181012180214"
    },
    "authoredBy": {
        "orcid": "https://orcid.org/0000-0001-9842-9718",
        "name": "Stian Soiland-Reyes"
    },

    "aggregates": [
        {
            "uri": "urn:hash::sha1:327fc7aedf4f6b69a42a7c8b808dc5a7aff61376",
            "bundledAs": {
                "uri": "arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/data/32/327fc7aedf4f6b69a42a7c8b808dc5a7aff61
                "folder": "/data/32/",
                "filename": "327fc7aedf4f6b69a42a7c8b808dc5a7aff61376"
            }
        },
        {
            "uri": "urn:hash::sha1:97fe1b50b4582cebc7d853796ebd62e3e163aa3f",
            "bundledAs": {
```

Consuming CWLProv ROs

cwlprov 0.1.1

```
pip install cwlprov
```

Latest version

Last released: About 3 days ago

cwlprov API for Python

https://pypi.org/project/cwlprov/

Navigation

Project description

Release history

Download files

Project links

Homepage

Statistics

Project description

CWLProv Python tool

The `cwlprov` Python tool is a command line interface to validate and inspect CWLProv Research Objects that capture workflow runs, typically executed in a Common Workflow Language implementation.

Installation

You'll need Python 3.

To install from pip try:

```
$ cwlprov --help
usage: cwlprov [-h] [--version] [--directory DIRECTORY] [--relative]
               [--absolute] [--output OUTPUT] [--verbose] [--quiet] [--hints]
               [--no-hints]
               {validate,info,who,prov,inputs,outputs,run,runs,rerun,derived,runtimes
               ...

cwlprov explores Research Objects containing provenance of Common Workflow
Language executions. <https://w3id.org/cwl/prov/>

commands:
{validate,info,who,prov,inputs,outputs,run,runs,rerun,derived,runtimes}
    validate            Validate the CWLProv Research Object
    info                show research object Metadata
    who                 show Who ran the workflow
    prov                export workflow execution Provenance in PROV format
    inputs              list workflow/step Input files/values
    outputs             list workflow/step Output files/values
    run                 show workflow Execution log
    runs                List all workflow executions in RO
    rerun               Rerun a workflow or step
    derived             list what was Derived from a data item, based on
                        activity usage/generation
    runtimes            calculate average step execution Runtimes
```

Validating

```
stain@biggie:~/src/cwlprov-py/test/revsort-cwlprov-0.4.0$ cwlprov --verbose validate
INFO:cwlprov.tool:Detected /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/bagit.txt
INFO:cwlprov.tool:Opening BagIt /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/data/32/327fc7aedf4f6b69a42a7c8b808dc5a7aff61376
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/data/97/97fe1b50b4582cebc7d853796ebd62e3e163aa3f
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/data/b9/b9214658cc453331b62c2282b772a5c063dbd284
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/workflow/packed.cwl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/workflow/primary-job.json
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.xml
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.provn
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.json
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.ttl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.nt
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/provenance/primary.cwlprov.jsonld
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/snapshot/revsort.cwl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/snapshot/hello.txt
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/snapshot/revtool.cwl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/snapshot/empty.ttl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/snapshot/sorttool.cwl
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/manifest.json
INFO:bagit:Verifying checksum for file /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/bag-info.txt
INFO:cwlprov.ro:Parsing RO manifest /home/stain/src/cwlprov-py/test/revsort-cwlprov-0.4.0/metadata/manifest.json
Valid CWLProv RO: .
```

https://github.com/fair-research/bdbag

Inspecting step runs

```
(venv3) stain@biggie:~/src/cwlprov-py/test/nested-cwlprov-0.3.0$ cwlprov run
2018-08-08 22:44:06.573330 Flow 39408a40-c1c8-4852-9747-87249425be1e [ Run of workflow/packed.cwl#main
2018-08-08 22:44:06.691722 Step 4f082fb6-3e4d-4a21-82e3-c685ce3deb58   Run of workflow/packed.cwl#main/create-tar  (0:00:00.01013
2018-08-08 22:44:06.702976 Step 0cceeaf6-4109-4f08-940b-f06ac959944a * Run of workflow/packed.cwl#main/compile  (unknown duration
2018-08-08 22:44:12.680097 Flow 39408a40-c1c8-4852-9747-87249425be1e ] Run of workflow/packed.cwl#main  (0:00:06.106767)
Legend:
[ Workflow start
* Nested provenance, use UUID to explore: cwlprov run 0cceeaf6-4109-4f08-940b-f06ac959944a
] Workflow end

(venv3) stain@biggie:~/src/cwlprov-py/test/nested-cwlprov-0.3.0$ cwlprov run 0cceeaf6-4109-4f08-940b-f06ac959944a
2018-08-08 22:44:06.607210 Flow 0cceeaf6-4109-4f08-940b-f06ac959944a [ Run of workflow/packed.cwl#main
2018-08-08 22:44:06.707070 Step 83752ab4-8227-4d4a-8baa-78376df34aed   Run of workflow/packed.cwl#main/untar  (0:00:00.008149)
2018-08-08 22:44:06.718554 Step f56d8478-a190-4251-84d9-7f69fe0f6f8b   Run of workflow/packed.cwl#main/argument  (0:00:00.532052)
2018-08-08 22:44:07.251588 Flow 0cceeaf6-4109-4f08-940b-f06ac959944a ] Run of workflow/packed.cwl#main  (0:00:00.644378)
Legend:
[ Workflow start
] Workflow end
```

```
stain@biggie:~/src/cwlprov-py/test/nested-cwlprov-0.3.0$ cwlprov outputs 4f082fb6-3e4d-4a21-82e3-c685ce3deb58 --format=files
Output tar:
data/c0/c0fd5812fe6d8d91fef7f4f1ba3a462500fce0c5

stain@biggie:~/src/cwlprov-py/test/nested-cwlprov-0.3.0$ tar tfv `cwlprov -q outputs 4f082fb6-3e4d-4a21-82e3-c685ce3deb58 --forma
-rw-r--r-- stain/stain     115 2018-08-08 23:44 Hello.java
```

```
stain@biggie:~/dropbox/work/cwlprov/newest-swift$ du -hs *
13G     alignment_0.6.0_linux
2.9G    rnaseqwf_0.5.0_mac
80M     somaticwf_0.5.0_ma
```

Feed Library Suggest Groups Datasets Careers Funding

Find Research Data My Datasets My Projects New Dataset FAQ

Title

CWL run of Somatic Variant Calling Workflow (CWLProv 0.5.0 Research Object)

Contributors

Contributor(s): Stian Soiland-Reyes | Farah Zaib Khan | + Add

Data files

Upload Files Upload Folder New Folder

somaticwf_0.5.0_mac Description Cite

CWLProv research object
See https://w3id.org/cwl/prov/0.5.0 or use https://pypi.org/project/cwlprov/ to explore
Note that Mendeley Data does not currently preserve folder structure if you do "Download all files".

data	+ Description	Cite	
metadata	+ Description	Cite	
snapshot	+ Description	Cite	
workflow	+ Description	Cite	
bag-info .txt	+ Description	0.28 KB	Cite
bagit .txt	+ Description	0.05 KB	Cite
manifest-sha1 .txt	+ Description	36 KB	Cite
tagmanifest-sha1 .txt	+ Description	10 KB	Cite

Draft (of version 2)

Last saved 2 days ago

Visibility: **Private**

Reserved DOI:
doi:10.17632/97hj93mkfd.2

Cite this dataset

Soiland-Reyes, Stian; Khan, Farah Zaib (2018), "CWL run of Somatic Variant Calling Workflow (CWLProv 0.5.0 Research Object)", Mendeley Data, v2

http://dx.doi.org/10.17632/97hj93mkfd.2

DOI is reserved but not active

Previous versions

Version 1 (unavailable) 2018-10-27

Licence

MIT Change

Storage available

100 GB

Space available: 99.90 GB

hared data Group

+ Add project data source

Stian's Dropbox

- cwlprov
  - 0.4.0-gdrive
  - newest-swift
  - incoming
  - zipped
    - rnaseqwf_0.5.0_mac.zip.sha256
    - somaticwf_0.5.0_mac.zip
    - somaticwf_0.5.0_mac.zip.sha256
    - rnaseqwf_0.5.0_mac.zip
    - alignment_0.6.0_linux.zip
    - alignment_0.6.0_linux.zip.sha256

Shared datasets Group

Your shared datasets

- CWL run of Alignment Workflow (CWLProv 0.6.0 Research Object)
- CWL run of RNAseq Workflow (CWLProv 0.5.0 Research Object)
- CWL run of Somatic Variant Calling Workflow (CWLProv 0.5.0 Research Object)

+ Add a new shared dataset

TODO: Best practice for publishing CWLProv ROs

>1 GB -> "Big Data"

What next for CWLProv?

Adding CWLProv support to Toil

Pau Ruiz Safont

EMBL-EBI

Toil

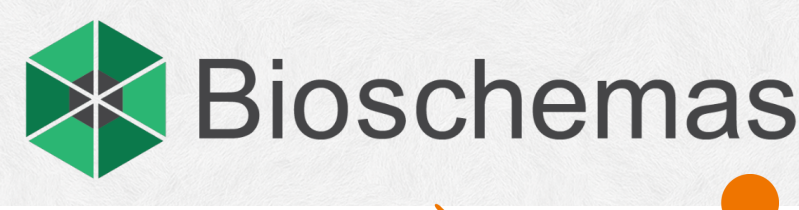
A scalable, efficient, cross-platform pipeline management syste

designed around the principles of functional programming.

Collect detailed usage statistics from compute environment

```
<?xml version="1.0" encoding="UTF-8"?>
<ur:UsageRecord xmlns="http://schema.ogf.org/urf/2013/04/urf"
        xmlns:ur="http://schema.ogf.org/urf/2013/04/urf" xmlns:xsi="http
        xsi:schemaLocation="http://schema.ogf.org/urf/2013/04/urf">
        <ur:RecordIdentityBlock>
                <ur:RecordId>urn:uuid:4350d583-61a5-45e8-a229-957aa81e80
                <ur:CreateTime>2018-05-09T09:06:52Z</ur:CreateTime>
                <ur:Site>EMBL-EBI</ur:Site>
                <ur:Infrastructure>Embassy</ur:Infrastructure>
        </ur:RecordIdentityBlock>
        <ur:SubjectIdentityBlock>
                <ur:LocalUserId>stain</ur:LocalUserId>
                <ur:LocalGroupId>ELIXIRCWLImplStudy</ur:LocalGroupId>
                <ur:GlobalUserId>https://orcid.org/0000-0001-9842-9718</
        </ur:SubjectIdentityBlock>
        <ur:ComputeUsageBlock>
                <ur:CpuDuration>PT3600S</ur:CpuDuration>
                <ur:WallDuration>PT3600S</ur:WallDuration>
                <ur:StartTime>2018-05-31T11:00:00</ur:StartTime>
                <ur:EndTime>2018-05-31T12:00:00</ur:EndTime>
                <ur:ExecutionHost>
                        <ur:Hostname>compute-0-1.example.com</ur:Hostnam
                        <ur:ProcessId>1042</ur:ProcessId>
                        <ur:Benchmark ur:type="si2k">3.14</ur:Benchmark>
                </ur:ExecutionHost>
                <ur:Processors>4</ur:Processors>
                <ur:NodeCount>1</ur:NodeCount>
```

Level 3: Domain-specific workflow annotations




```yaml
outputs:
  sequence:
    type: stdout
  format: edam:format_1929 # FASTA
  cwlprov:concept: ebi_metagenomics:assembly_statistics
  cwlprov:relationships:
      prov:wasDerivedFrom: [ inputs.second_input, outputs.second_output ]
```

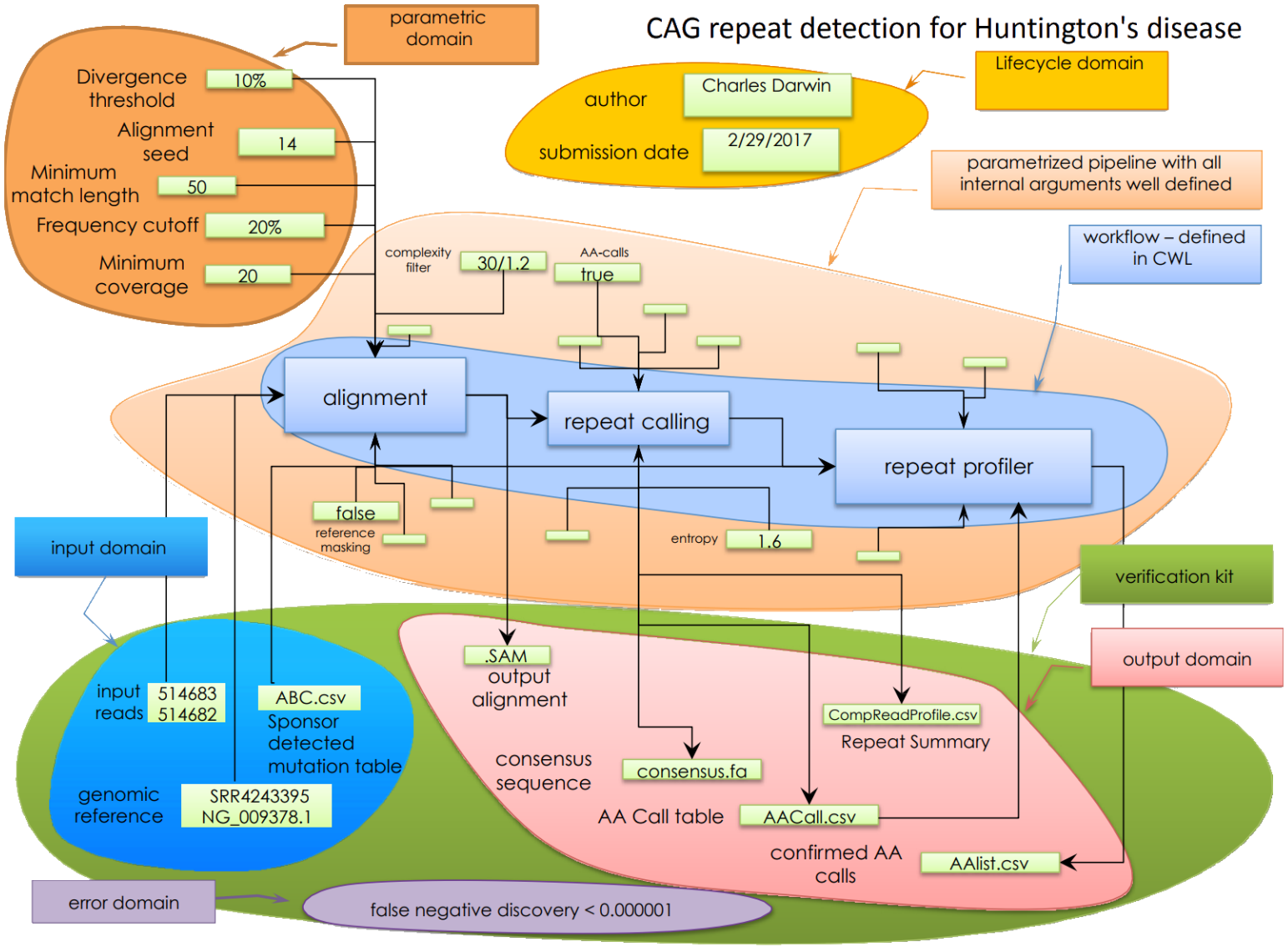
postprocessing of Research Object

```
{
    "uri": "urn:hash::sha1:b9214658cc453331b62c2282b772a5c063dbd284",
    "bundledAs": {
        "uri": "arcp://uuid,1f767ad4-ac52-4623-b5bc-dd9faf2b869f/data/b9/b9214658cc453331b62c2282b772a5c063dbd284"
        "folder": "/data/b9/",
        "filename": "b9214658cc453331b62c2282b772a5c063dbd284"
    },
    "format": "http://edamontology.org/format_1929",
    "classifying":  "ebi_metagenomics:assembly_statistics",
    "prov:wasDerivedFrom": [
       "urn:hash::sha1:f572d396fae9206628714fb2ce00f72e94f2258f",
       "urn:hash::sha1:2258ff572d396fae9206628714fb2ce00f72e94f",
    ]
},
```

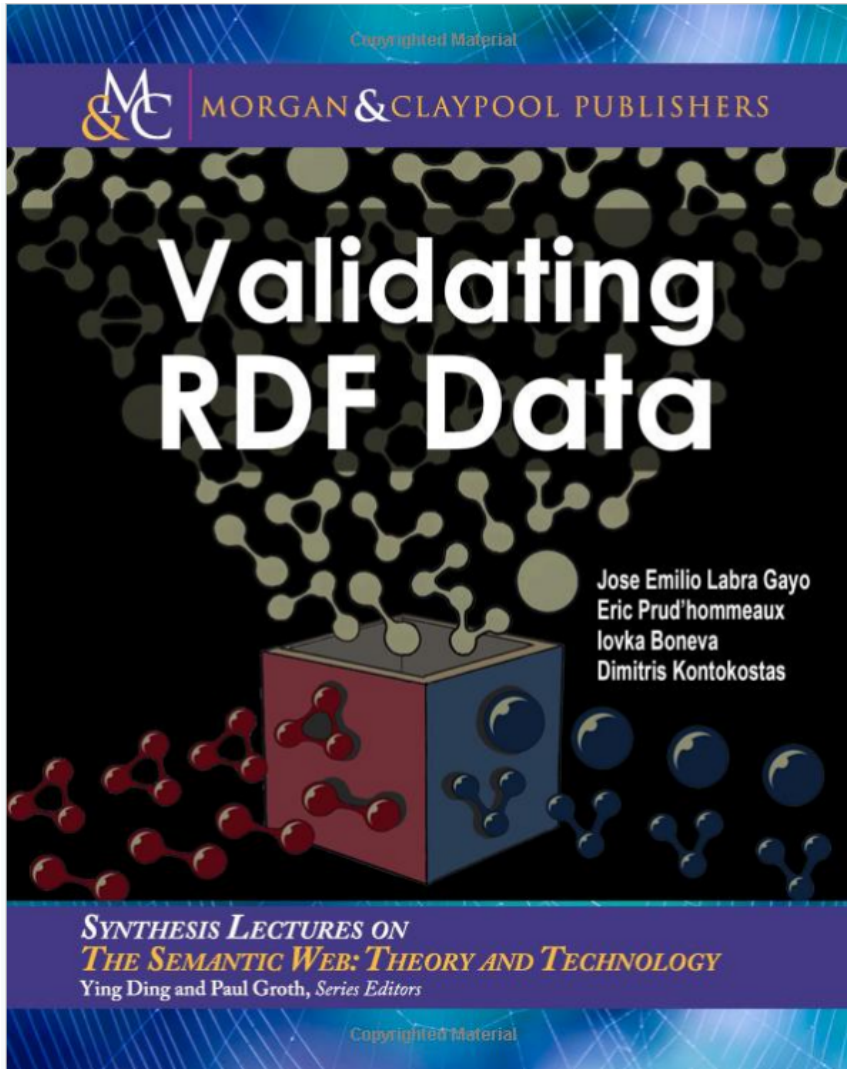
BioCompute Objects

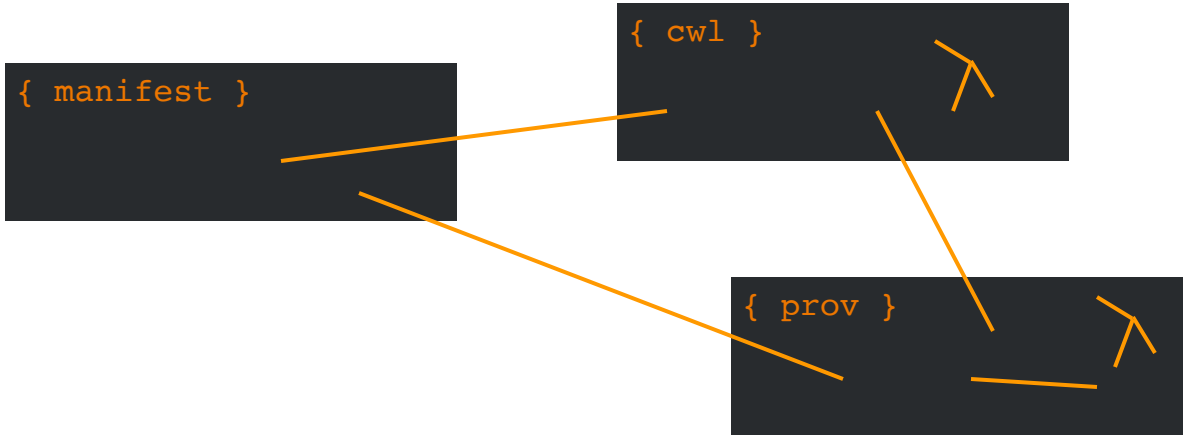
https://github.com/biocompute-objects/BCO_Specification

```
outputs:
  sequence:
    type: stdout
  biocompute:error_domain: [
    "frequency_cutoff > 0.05"
  ],
  biocompute:input_domain: {
    "genomic_reference": "WAP_RAT"
  }
```



Deep Validation





Adam Cowdy

MANCHESTER 1824
The University of Manchester

MENDELEY DATA

NIH Data Commons Pilot Phase Explores Using the Cloud to Access and Share *FAIR Biomedical Big Data

***F**indable, **A**ccessible, **I**nteroperable and **R**eusable

Open for Innovation
KNIME

CWLProv without CWL..?

PAPER

CWLProv: Interoperable Retrospective Provenance Capture and Computational Analysis Sharing

Farah Zaib Khan[1,2,*], Stian Soiland-Reyes[2,3,*], Richard O. Sinnott[1,*], Andrew Lonie[1,*], Carole Goble[3,*] and Michael R. Crusoe[2,*]

[1]The University of Melbourne, Australia and [2]Common Workflow Language Project and [3]The University of Manchester

[*]khanf1@unimelb.edu.au; soiland-reyes@manchester.ac.uk; rsinnott@unimelb.edu.au; alonie@unimelb.edu.au; carole.goble@manchester.ac.uk; mrc@commonwl.org

Abstract

Background: The automation of data analysis in the form of *scientific workflows* has become a widely adopted practice in many fields of research. Computationally driven data-intensive experiments using workflows enable Automation, Scaling, Adaption and Provenance support (ASAP). However, there are still several challenges associated with the effective sharing, publication, understandability and reproducibility of such workflows due to the incomplete capture of provenance and lack of interoperability between different technical (software) platforms.

Results: Based on best practice recommendations identified from literature on workflow sharing and publishing, we define four hierarchical levels of provenance that collectively result in comprehensive and fully re-executable workflows when used with domain-specific information. To realise these levels, we present *CWLProv*, a standard-based format to represent any workflow-based computational analysis to produce workflow output artefacts that satisfy the various levels of provenance. We utilise open source community-driven standards; interoperable workflow definitions in Common Workflow Language (CWL), structured provenance representation using the W3C PROV model, and resource aggregation and sharing as workflow-centric Research Objects (RO) generated along with the final outputs of a given workflow enactment. We illustrate this approach through a practical demonstration of *CWLProv* applied to real-life genomic workflows developed by independent groups.

Conclusions: Our approach to workflow sharing and publication mitigates *workflow decay*. The underlying principles of the standards utilised by *CWLProv* enable semantically-rich and executable Research Objects that capture computational workflows with retrospective provenance such that any platform supporting CWL will be able to understand the analysis, re-use the methods for partial re-runs, or reproduce the analysis to validate the published findings.

Key words: Provenance; Common Workflow Language; CWL; Research Object; RO; BagIt; Interoperability; Portability; Scientific Workflows; Containers

Introduction

Out of the many big data domains, genomics is considered "*the most demanding*" with respect to all stages of the data lifecycle - from acquisition, storage, distribution and analysis [1]. As genomic data is growing at an unprecedented rate due to improved sequencing technologies and reduced cost, it is currently challenging to analyse the data at a rate matching its production. With data growing exponentially in size and volume, the practice to perform computational analyses using *workflows* has overtaken more traditional research methods using ad-hoc scripts that has been the typical modus operandi over the last few decades [2, 3]. Scientific workflow design and management has become an essential part of many computationally

Compiled on: October 28, 2018.
Draft manuscript prepared by the author.

COMMON WORKFLOW LANGUAGE
https://www.commonwl.org/

researchobject.org
http://view.commonwl.org/

http://w3id.org/cwl/prov

https://doi.org/10.5281/zenodo.1473157

bioexcel
https://bioexcel.eu/workflows/