RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

Research data management

Workflows with the Common Workflow Language

Christoph Garth

Motivation

- Research data should be FAIR
  - Annotation, metadata, ontologies, ...
- What about computational aspects of the research process?
  - Reproducability?
  - Reusability?
  - Annotation?
  - Search?

Workflows

- Workflow[1]: roughly

a collection of computer applications, scripts, and code, used in computational data analysis

  - which programs are used
  - how they are configured
  - how the data flows between them

[1] (primarily in a research/scientific context)

Workflow Management Systems

... offer facilities for

- **describing** workflows
  - programs / scripts, parameters, inputs / outputs (data dependencies)
  - environment (e.g. packages, container, RAM, cores)
- **excuting** workflows
  - on your PC
  - on a cluster (e.g. Elwetritsch)
  - on some other kind of infrastructure (e.g. de.nbi, AWS, Azure)
- Computational Workflow Management Systems (cWMS): focus on scientific computations

Why use a cWMS?

Features beyond portability and reproducability include:

- Separation of concerns
  - focus on science first, then optimize execution later
- Automated execution
  - Start a complex analysis involving many pieces with a single command
- Scaling
  - Across nodes, clusters, continents...
- Automatic provenance
  - How was this file made?

Existing cWMS

Over 300 cWMS

see
https://github.com/common-workflow-language/common-workflow-language/wiki/Existing-Workflow-systems

1. Arvados - CWL-based distributed computing platform for data analysis on massive data sets. https://arvados.org/ https://github.com/arvados/arvados
2. Apache Taverna http://www.taverna.org.uk/ https://taverna.incubator.apache.org/
3. Galaxy http://galaxyproject.org/
4. SHIWA https://www.shiwa-workflow.eu/
5. Apache 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/~~-archived at https://web.archive.org/web/20180609011656/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/
12. make, rake, drake, ant, scons & many others. Software development relies heavily on tools to manage workflows related to compiling and packaging applications. For the most part these are file based and usually run on a single node, usually supporting parallel steps (make -j) and in some cases able to dispatch build steps to other machines https://code.google.com/p/distcc/ https://github.com/Factual/drake
13. Snakemake https://snakemake.github.io
14. BPipe http://bpipe.org http://docs.bpipe.org/
15. Ruffus https://github.com/cgat-developers/ruffus
16. NextFlow http://nextflow.io
17. Luigi. Python package that helps you build complex pipelines of batch jobs http://github.com/spotify/luigi https://luigi.readthedocs.io
18. SciLuigi. Helper library built on top of Luigi to ease development of Scientific workflows in Luigi: http://github.com/pharmbio/sciluigi
19. Luigi Analysis Workflow (LAW) https://github.com/riga/law
20. GATK Queue https://www.broadinstitute.org/gatk/guide/topic?name=queue
21. Yabi https://ccg.murdoch.edu.au/yabi
22. seqware Workflows are written in Java and executed using the Oozie Workflow Engine on Hadoop or SGE clusters. Uses Zip64 files to group the workflow definition file, workflow itself, sample settings, and data dependencies in a single file that can be exchanged between SeqWare users or archived. https://seqware.github.io/ https://seqware.github.io/docs/6-pipeline/
23. Ketrew https://github.com/hammerlab/ketrew
24. Pegasus http://pegasus.isi.edu/
25. Apache Airflow https://github.com/apache/airflow
26. Couler https://github.com/couler-proj/couler - Unified interface for constructing and managing workflows on different workflow engines, such as Argo Workflows, Tekton Pipelines, and Apache Airflow.
27. Cosmos https://cosmos.hms.harvard.edu/documentation/index.html http://bioinformatics.oxfordjournals.org/content/early/2014/07/24/bioinformatics.btu385.full [paper] Cosmos2: https://github.com/LPM-HMS/COSMOS2 http://cosmos.hms.harvard.edu/COSMOS2/
28. Pinball https://github.com/pinterest/pinball
29. bcbio https://bcbio-nextgen.readthedocs.org/en/latest/
30. Chronos https://github.com/mesos/chronos
31. Azkaban https://azkaban.github.io/
32. Apache NiFi https://nifi.apache.org/docs/nifi-docs/html/overview.html
33. flowr (R-based) http://docs.flowr.space/ https://github.com/sahilseth/flowr
34. Mistral https://github.com/arteria-project https://wiki.openstack.org/wiki/Mistral#What_is_Mistral.3F https://docs.openstack.org/mistral/latest/user/wf_lang_v2.html
35. nipype http://nipy.org/nipype/
36. End of Day https://github.com/joestubbs/endofday
37. BioDSL https://github.com/maasha/BioDSL
38. BigDataScript http://pcingola.github.io/BigDataScript/
39. Omics Pipe: uses Ruffus http://sulab.scripps.edu/omicspipe/
40. Ensembl Hive https://github.com/Ensembl/ensembl-hive
41. QuickNGS http://bifacility.uni-koeln.de/quickngs/web
42. GenePattern http://www.broadinstitute.org/cancer/software/genepattern/
43. Chipster http://chipster.csc.fi/
44. The Genome Modeling System https://github.com/genome/gms
45. Cuneiform, A Functional Workflow Language https://github.com/joergen7/cuneiform http://www.cuneiform-lang.org/
46. Anvaya http://www.ncbi.nlm.nih.gov/pubmed/22809419 http://webapp.cabgrid.res.in/biocomp/Anvaya/ANVAYA_Main.html#HOWTO_INSTALL_ANVAYA
47. Makeflow http://ccl.cse.nd.edu/software/makeflow/
48. Apache Airavata http://airavata.apache.org/
49. Pyflow https://github.com/Illumina/pyflow
50. Cluster Flow http://clusterflow.io
51. Unipro UGENE http://ugene.net/ https://doi.org/10.7717/peerj.644
52. CloudSlang http://www.cloudslang.io/
53. Stacks http://catchenlab.life.illinois.edu/stacks/
54. Leaf http://www.francesconapolitano.it/leaf/index.html
55. omictools http://omictools.com/
56. Job Description Language. The Job Description Language, JDL, is a high-level, user-oriented language based on Condor classified advertisements for describing jobs and aggregates of jobs such as Direct Acyclic Graphs and Collections. https://edms.cern.ch/ui/file/590869/1/WMS-JDL.pdf
57. YAWL yet another workflow language https://doi.org/10.1016/j.is.2004.02.002 http://www.yawlfoundation.org/

Why have a standard at all?

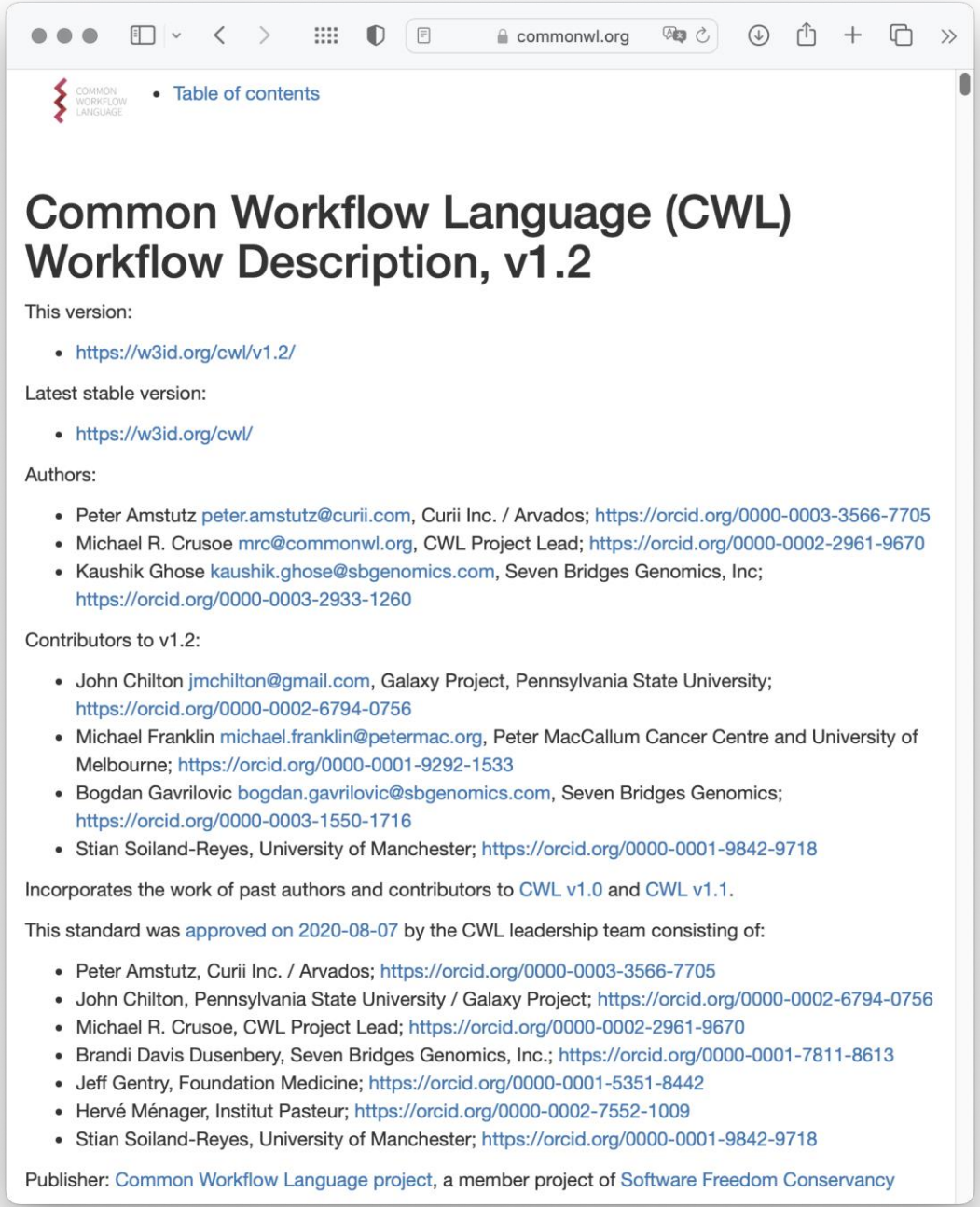
- Standards create a surface for collaboration and promote innovation
- Research frequently dips in and out of different systems but interoperability is not a basic feature
- Funders, journals, and other sources of incentives prefer standards over proprietary or single-source approaches

Common Workflow Language

- Common format for computational tool & workflow execution (initially bioinformatics, but not limited to that)
- Extensible, community-based standard, not a specific software package or platform
  - Schema-based definition, specification and test suite
- Aimed at: local execution, shared-nothing clusters, academic clusters, cloud environments, supercomputers
- Supports use of containers (e.g. Docker)

CWL 1.2

- Approved by community process in 2020



commonwl.org

- Table of contents

Common Workflow Language (CWL) Workflow Description, v1.2

This version:

- https://w3id.org/cwl/v1.2/

Latest stable version:

- https://w3id.org/cwl/

Authors:

- Peter Amstutz peter.amstutz@curii.com, Curii Inc. / Arvados; https://orcid.org/0000-0003-3566-7705
- Michael R. Crusoe mrc@commonwl.org, CWL Project Lead; https://orcid.org/0000-0002-2961-9670
- Kaushik Ghose kaushik.ghose@sbgenomics.com, Seven Bridges Genomics, Inc; https://orcid.org/0000-0003-2933-1260

Contributors to v1.2:

- John Chilton jmchilton@gmail.com, Galaxy Project, Pennsylvania State University; https://orcid.org/0000-0002-6794-0756
- Michael Franklin michael.franklin@petermac.org, Peter MacCallum Cancer Centre and University of Melbourne; https://orcid.org/0000-0001-9292-1533
- Bogdan Gavrilovic bogdan.gavrilovic@sbgenomics.com, Seven Bridges Genomics; https://orcid.org/0000-0003-1550-1716
- Stian Soiland-Reyes, University of Manchester; https://orcid.org/0000-0001-9842-9718

Incorporates the work of past authors and contributors to CWL v1.0 and CWL v1.1.

This standard was approved on 2020-08-07 by the CWL leadership team consisting of:

- Peter Amstutz, Curii Inc. / Arvados; https://orcid.org/0000-0003-3566-7705
- John Chilton, Pennsylvania State University / Galaxy Project; https://orcid.org/0000-0002-6794-0756
- Michael R. Crusoe, CWL Project Lead; https://orcid.org/0000-0002-2961-9670
- Brandi Davis Dusenbery, Seven Bridges Genomics, Inc.; https://orcid.org/0000-0001-7811-8613
- Jeff Gentry, Foundation Medicine; https://orcid.org/0000-0001-5351-8442
- Hervé Ménager, Institut Pasteur; https://orcid.org/0000-0002-7552-1009
- Stian Soiland-Reyes, University of Manchester; https://orcid.org/0000-0001-9842-9718

Publisher: Common Workflow Language project, a member project of Software Freedom Conservancy

Design principles

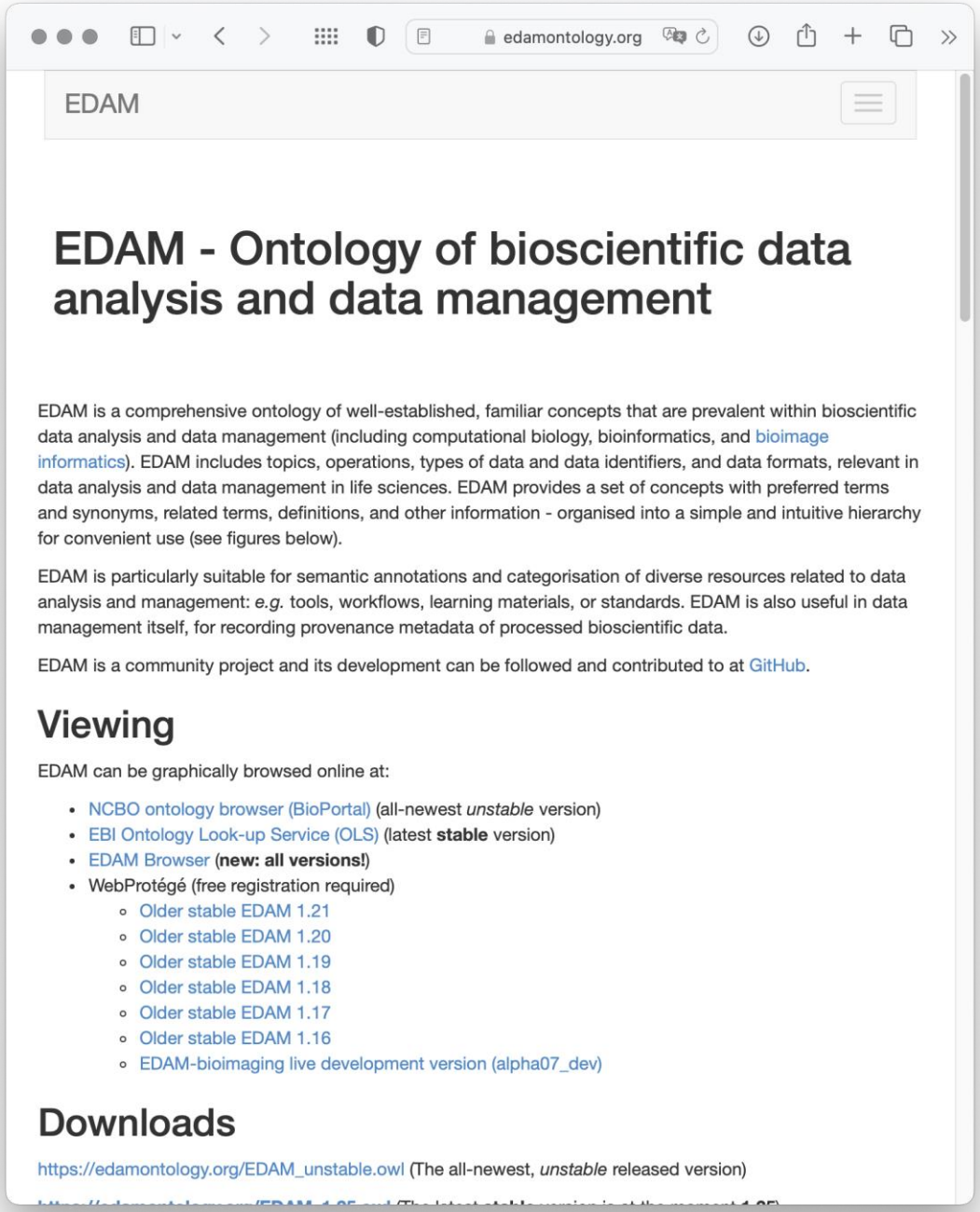
- Low barrier to entry for users and implementers
- Convenience through tooling (GUIs, converters)
- Extensions allowed (well-marked)
- Part of the linked-data ecosystem
  (cf. RO Crate presentation yesterday)

- PRAGMATIC
  prioritize usefulness over theoretical purity

Linked Data & CWL

- Hyperlinks are common
- Bring your own ontologies (RDF) for metadata

Example: can use the EDAM ontology to specify file formats



edamontology.org

EDAM

EDAM - Ontology of bioscientific data analysis and data management

EDAM is a comprehensive ontology of well-established, familiar concepts that are prevalent within bioscientific data analysis and data management (including computational biology, bioinformatics, and bioimage informatics). EDAM includes topics, operations, types of data and data identifiers, and data formats, relevant in data analysis and data management in life sciences. EDAM provides a set of concepts with preferred terms and synonyms, related terms, definitions, and other information - organised into a simple and intuitive hierarchy for convenient use (see figures below).

EDAM is particularly suitable for semantic annotations and categorisation of diverse resources related to data analysis and management: *e.g.* tools, workflows, learning materials, or standards. EDAM is also useful in data management itself, for recording provenance metadata of processed bioscientific data.

EDAM is a community project and its development can be followed and contributed to at GitHub.

Viewing

EDAM can be graphically browsed online at:

- NCBO ontology browser (BioPortal) (all-newest *unstable* version)
- EBI Ontology Look-up Service (OLS) (latest **stable** version)
- EDAM Browser (**new: all versions!**)
- WebProtégé (free registration required)
  - Older stable EDAM 1.21
  - Older stable EDAM 1.20
  - Older stable EDAM 1.19
  - Older stable EDAM 1.18
  - Older stable EDAM 1.17
  - Older stable EDAM 1.16
  - EDAM-bioimaging live development version (alpha07_dev)

Downloads

https://edamontology.org/EDAM_unstable.owl (The all-newest, *unstable* released version)

Why not containers / shell scripts / Slurm?

Containers:

- Solve portability, but not execution and data flow
- Lack of composability: home-grown mega-containers
- Cannot work well without „driver“ scripts

Shell Scripts (== Slurm):

- Not (easily) scalable, not (easily) portable, not (easily) reproducable, ...

Example: Bioinformatics Pipelines

https://www.ebi.ac.uk/metagenomics/pipelines/3.0

data analysis workflows for metagenomics

ca. 10k lines of Python / Bash / Perl code

⬇

2.5k lines of CWL descriptions

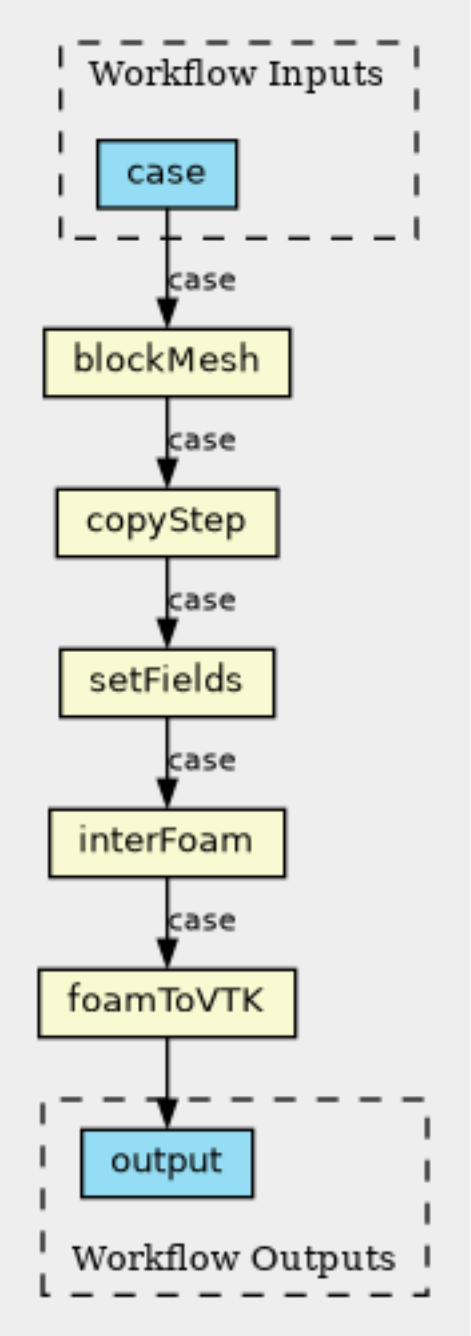
https://github.com/EBI-Metagenomics/ebi-metagenomics-cwl

Example

CWL-OpenFOAM

Can run

- on local machine using **cwltool** and **Docker**
- on HPC cluster using cWMS **Toil** and **Singularity**.



CommandLineTool

CWL description of a single tool in YAML[1] format:

- which command to run
- inputs
- outputs

```yaml
cwlVersion: v1.2
class: CommandLineTool

baseCommand: /usr/games/cowsay

requirements:
  DockerRequirement:
    dockerPull: chuanwen/cowsay

inputs:
  message:
    type: string
    inputBinding:
      position: 1

outputs:
  cowfile:
    type: File
    outputBinding:
      glob: output.txt
```

1 https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started)

InputParameter

Specify name and type of input parameters

- type can be null, boolean, int, string, float, array, record
- inputBinding describes how parameter is turned into command line argument

```yaml
inputs:
  message:
    type: string
    inputBinding:
      position: 1
```

Outputs

- Specify name and type of output parameters, e.g.
  - (list of) files
  - (list of)directories
  - strings,
  - ...

```yaml
outputs:
  cowfile:
    type: File
    outputBinding:
      glob: output.txt
```

- Often: capture stdout

Execution

- Using any CWL runtime, e.g. cwltool

```
> cwltool cwl/cowsay.cwl --message "I like CWL"
```

- Parameters can be given in dedicated YAML file (input object)

```
> cat cowsay.yaml
message: I like CWL very much!
> cwltool cwl/cowsay.cwl cowsay.yaml
```

Workflow

Graph of processing steps

in YAML format

- per step, run tool or workflow
- connect step inputs and outputs

```yaml
cwlVersion: v1.2
class: Workflow

inputs: []

steps:
  step1:
    run: fortune.cwl
      in: []
      out: [message]
  step2:
    run: cowsay.cwl
    in:
      message: step1/message
    out: [cowfile]

outputs:
  out:
  type: File
    outputSource: step2/cowfile
```

1 https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started)

Metadata

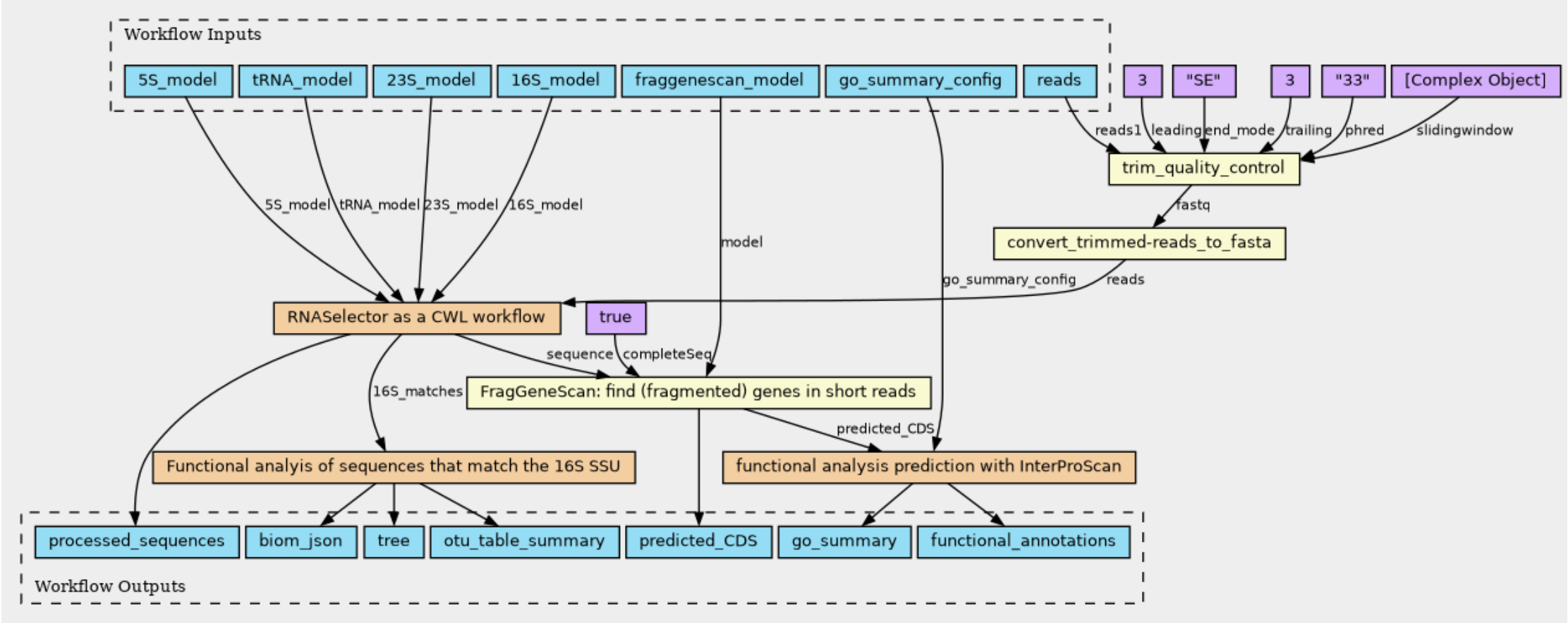
CWL is extensible via the schema salad mechanism

- can annotate arbitrary metadata if there is a schema

```
s:author:
- class: s:Person
  s:identifier: https://orcid.org/0000-0003-1669-8549
  s:email: mailto:garth@rptu.de
  s:name: Christoph Garth

s:contributor:
- class: s:Person
  s:identifier: http://orcid.org/0000-0003-3925-6778
  s:email: mailto:timo.muehlhaus@rptu.de
  s:name: Timo Mühlhaus

s:license: https://spdx.org/licenses/GPL-3.0-or-later

$namespaces:
  s: https://schema.org/

$schemas:
  - https://schema.org/version/latest/schemaorg-current-https.rdf
```

Free Stuff! Visualization

Plug URL into workflow to https://view.commonwl.org



Challenges

- CWL is limited
  - no looping or conditionals; not a full programming language
  - but typically not needed for macroscale workflows
- CWL is an evolving standard in an open standards ecosystem
  - community-driven frequent releases
  - now an ISA process description
- CWL is not the only cWMS language
  - NextFlow, snakeMake, ...
- YMMV...
  - while implementations mature (esp. HPC + portability)

Take Home

- Prefer cWMS and workflow languages over home-grown shell scripts
  - Standardized, Portable
  - FAIRer by default
- Go for lightweight, incremental adoption
  - Don‘t port everything now, but maybe adopt for new projects