Capturing interoperable reproducible workflows

DOI:
10.5281/zenodo.1312623

Document Version
Accepted author manuscript

Link to publication record in Manchester Research Explorer

Citation for published version (APA):
https://doi.org/10.5281/zenodo.1312623

Citing this paper
Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights
Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy
If you believe that this document breaches copyright please refer to the University of Manchester’s Takedown Procedures [http://man.ac.uk/04Y6Bo] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.
Capturing interoperable reproducible workflows with Common Workflow Language

Identifier
https://doi.org/10.5281/zenodo.1312623

Date created
2018-07-16

Submitted to
Workshop on Research Objects (RO2018)

Authors
Stian Soiland-Reyes The University of Manchester, UK; Common Workflow Language project; Apache Taverna (incubating)
https://orcid.org/0000-0001-9842-9718

Farah Zaib Khan The University of Melbourne, Australia; Common Workflow Language project
https://orcid.org/0000-0002-6337-3037

Richard O. Sinnott The University of Melbourne, Australia
https://orcid.org/0000-0001-5998-222X

Andrew Lonie The University of Melbourne, Australia
https://orcid.org/0000-0002-2006-3856

Michael R Crusoe Common Workflow Language project
https://orcid.org/0000-0002-2961-9670

Carole Goble The University of Manchester, UK
https://orcid.org/0000-0003-1219-2137

Abstract
We present our ongoing work on integrating Research Object practices with Common Workflow Language, capturing and describing prospective and retrospective provenance.

Background
Performing computational data analysis using scientific workflows is common in many domains, including bioinformatics, physics, earth sciences and cheminformatics [1]; and it is now considered best practice to use workflows or automated scripts to improve reproducibility of analyses [2]. Recent advances in software containers and packaging help remove previous challenges in capturing software tools needed by an analysis [3,4], but other interoperability factors remain important, such as ensuring a pipeline definition can execute in multiple workflow systems [5], and that provenance of workflow runs can be reliably captured and used across heterogeneous execution environments [6]. While robust reproducibility of results is important, the outcomes and methods also need to be replicable and generalizable [7].

Common Workflow Language (CWL) [8] is a multivendor community-led standardization effort to define a dataflow language that is implemented by multiple workflow systems capable of executing on a wide range of compute platforms. CWL focus primarily on what has been determined as a common denominator across bioinformatics pipeline systems; coordinating and parallelizing command line tools that exchange files. While CWL started in the bioinformatics community, usage and adoption has spread beyond the life sciences to radio astronomy, hydrology, digital humanities, and more.
CWL has seen considerable interest over recent years [9], in particular primed for the role of computational interoperability, as FAIR sharing of data [10] should also include the workflows that produced it [11].

**Prospective provenance — capturing the workflow template**

We created CWL Viewer [12] for visualizing CWL workflow definitions, and it has now more than 2500 CWL workflows registered. Most of these are developed as open source in GitHub repositories. While continual active development of workflows is good for reuse and stability, it adds a challenge for reproducibility and reliable workflow citation. CWL workflows are typically split in multiple files (e.g. one per command line tool), and may contain internal references used during execution, such as secondary data files. Although CWL encourages interoperability, it has not yet defined a standard mechanism to reliably capture or transfer the complete definition required for starting a workflow run. A given CWL workflow and its components can be richly annotated using vocabularies like EDAM [13] and schema.org to indicate formats, contributors, licenses and references, but, while this reuse follows Linked Data principles, the metadata is embedded within the CWL definitions, requiring Schema Salad preprocessing before further use.

To this end we have extended the CWL Viewer to produce permalinks based on the corresponding git commit, adding content-negotiation to other formats (SVG, RDF Turtle, JSON-LD) with archiving as Research Objects [14] that include snapshots of the CWL files, their dependent resources and extracted metadata augmented with authorship information extracted from the git log. This work aims to capture the prospective provenance of a workflow definition; the metadata that should be true for every execution.

To complete this picture we are working with Open AIRE to make CWL workflows Findable, and plan to calculate prospective execution details such as linking Docker images [15] and BioConda packages [16] with SciCrunch RRIDs [17] and ELIXIR’s Tools and Data Services registry [18] using CWL SoftwarePackage annotations.

**Retrospective provenance — how was the workflow executed?**

In terms of retrospective provenance, capturing what happened in a particular execution of a CWL workflow, we have been extending our earlier work on capturing workflow runs as research objects [14]. Previous work with TavernaProv [19] and Wings [20] both used the distinction of prospective vocabularies with wfdesc [14] or P-Plan [21] of the workflow template, and retrospective PROV statements with wfprov [14] or OPMW [22] of the instantiated workflow execution; thus allowing queries and views of the abstracted workflow independent of its execution, however computational re-execution or reuse of the workflows were effectively still restricted to their original workflow engines.

CWL is implemented by multiple workflow engines, with different execution characteristics (e.g. distributed on cloud), but with an explicit and interoperable definition of what is to be run. Therefore the aim was to define a more specific Research Object profile for workflow execution that should enable more precise replication of the workflow run, but at the same time being general enough to be used by multiple engines.

For this we created CWLProv [23], developed by modifying the CWL reference implementation cwltool to add a new --provenance flag. The resulting profile describes how the existing Linked Data standards for Research Object, wfdesc, wfprov [14], PROV [24] and Web Annotation Data Model [25] are combined to track detailed workflow execution logs.

Unlike TavernaProv, which used Research Object Bundle ZIP files [26], CWLProv uses BagIt for archiving the files of the RO. BagIt [27] is a Library of Congress supported digital preservation format, that can exist either as a flat file hierarchy or packaged as tar/ZIP archives. BagIt support external references and checksums of all files, but its metadata provisions are fairly minimal.

Complementing the two approaches we use the BDBag profile for Research Object in BagIt, as it has previously been shown to support data from large-scale workflows [28,29] and is also of consideration by NIH.
The **PROV files captured by CWLProv** [23] reuse the wdesc and wfprov vocabularies, but compared to TavernaProv stay closer to plain PROV statements. In cwltool we used the **Prov Python** library and could thus save in multiple serializations, including PROV-N, PROV-XML, PROV-JSON as well as PROV-O based RDF as RDF Turtle, N-Triples and JSON-LD. It is still under investigation which of these formats will be most beneficial for consumers and producers, thus CWLProv only mandate PROV-N for now. CWLProv uses different *provenance levels* to avoid complicating coarse-grained provenance with the details of fine-grained execution traces.

The CWLProv approach is also the basis for Research Object support **being developed** for the workflow system Nextflow [31], but there without requiring an executable CWL workflow. One interesting collaboration aspect we are exploring is to identify common approaches not previously described for Workflow Research Objects, such as capturing execution of Docker images or referencing large data files.

**References**


Acknowledgements

FZK funded by Melbourne International Research Scholarship (MIRS) and Melbourne International Fee Remission Scholarship (MIFRS). SSR funded by BioExcel CoE, a project funded by the European Union contract H2020-EINFRA-2015-1-675728.