GigaScience, 2017, 1–19

doi: xx.xxxx/xxxx
Manuscript in Preparation
Research

RESEARCH

A Non-Intimidating Approach to Workflow Reproducibility in Bioinformatics: Adding Metadata to Research Objects through the Design and Evaluation of Use-Focused Extensions to CWLProv

Renske de Wit[1,2,*], Alexandru Iosup[2,†], Sanne Abeln[1,3,‡] and Michael R. Crusoe[1,2,§,¶]

[1]VU Bioinformatics Group, Dept. Computer Science, Vrije Universiteit Amsterdam and [2]Massivizing Computer Systems Group, Dept. Computer Science, Vrije Universiteit Amsterdam and [3]AI Technology for Life Group, Dept. Information and Computing Sciences and Dept. Biology, Utrecht University

[*]r.d.wit@nki.nl ; [†]a.iosup@vu.nl ; [‡]s.abeln@vu.nl ; [§]michael.crusoe@vu.nl
[¶]Corresponding author.

Renske: Change title? 'Defining an actionable provenance standard for research objects through...'? '...framework'?

Abstract

Background: Computational reproducibility is important for reuse of scientific research, but is often not achieved across many domains of science due to the complexity of data and analysis pipelines. Workflow-centric research objects (ROs) have been proposed as a machine-accessible format for sharing computational research, encapsulating data as well as metadata to describe the provenance of the results. However, due to the large variety of metadata standards it is challenging to prioritize a set of provenance metadata that is most useful in later stages of the research. Here, we present a systematic approach to define such a set of metadata, and demonstrate how this strategy can guide the improvement of CWLProv, an existing RO specification for Common Workflow Language (CWL) workflows. **Results**: Instead of analyzing many workflows in a particular domain, we chose a single exemplary bioinformatics workflow and studied it in detail. Grounded in five realistic use case scenarios for ROs associated with the workflow, we identified relevant provenance questions and defined the metadata required to answer them. Subsequently, we analyzed the current CWLProv community standard for the representation of this metadata, distinguishing different qualities of representation. Finally, we used the insights gained from our analysis to design two extensions to CWLProv, addressing provenance requirements that were insufficiently represented before. **Conclusions**: The method described here can aid workflow practitioners to identify the set of most useful metadata in their domain, and guide workflow system developers to implement automatic capture of provenance during workflow execution.

Key words: provenance; research object; computational reproducibility; metadata; workflow; provenance analytics

Michael: Abstract needs work

Background

In the era of big data and big science, *computational reproducibility* is an important means to verify the validity of scientific conclu-

Michael: needs work

Alex: Comment by Alex

Compiled on: December 12, 2023.
Draft manuscript prepared by the author.

Key Points

- Research objects (ROs) have been proposed as a machine-accessible format to capture the provenance of workflow executions, improving the reuse and reproducibility of computational research.
- Due to a wide range in metadata standards and ontologies, it has become difficult for practitioners and workflow system developers to prioritize which metadata should be collected to meet the provenance requirements of real workflows.
- Here, we analyze the provenance requirements of one bioinformatics workflow in detail, and show how the results can be used to inform the development of RO specifications like CWLProv.

sions drawn from raw data, and promotes *reuse* of the research in downstream analyses. However, across multiple domains of science, computational reproducibility is often not achieved due to lack of transparency or *code rot* [1]. Given the prominent role which science plays in our society, this is a serious and challenging problem. Manually collecting all the information required for reproducibility is tedious, since current data analysis requires complex computational processes which use large datasets from many different sources, comprise various software tools, and are often executed on various computational systems. Solutions which automate the recording of *provenance*, i.e. all the steps which led to a particular result, are therefore in high demand.

The transparency of computational analyses can be improved by expressing scientific analyses as *workflows*, i.e. a series of steps with the output of one step becoming the input for another. Figure 1 envisions a future in which workflows are an integral part of reproducible computational research, central to an ecosystem of tools and platforms which have emerged over the last decade. A crucial requirement to the success of this approach is that the tools are compatible with each other, i.e. *interoperable*.

The problem of interoperability has been partly addressed by the specification of Common Workflow Language (CWL) [2], a platform-neutral standard for workflow descriptions. CWL workflows are *portable* across multiple workflow management systems. However, provenance of a computational result comprises more than a workflow description alone, since the output data are also strongly dependent on the input configurations (data and parameter settings) and the (distributed) computational system(s) on which the workflow was executed.

Michael: Comment by Michael

Sanne: Comment by Sanne

For this reason, workflow-centric *research objects* (ROs) [3] have been proposed as a common, machine-accessible standard for the storage and distribution of computational results. ROs are collections of resources that together contribute to a scientific result, with semantic annotations describing the aggregated entities and their relations to each other.

CWLProv [4] is a specialization of the RO model for CWL workflows. CWLProv *RO Bundles* are created during execution by the CWL reference runner *cwltool* [5] and capture the workflow description, input and output data for each step, and a machine-accessible record of the execution in RDF [6], according to the W3C PROV-DM standard [7]. The tool *cwlprov-py* [8] can extract provenance metadata from the RO Bundle and rerun individual steps or the entire workflow.

Although CWLProv has great potential to improve computational reproducibility, the current version of the specification only captures what happens *during* workflow execution, while provenance metadata associated with the workflow inputs (if it exists) is lost. One possible solution to this problem is to ask CWL users to add semantic annotations to their input data and workflows, which is already encouraged in the CWL Standards. However, in reality, the wide variety in existing metadata standards and ontologies make it difficult to decide which annotations should be prioritized: the metadata captured in the RO should ultimately be able to fulfill the roles which provenance plays in practice, i.e. *provenance analytics* [9]. Defining and understanding these purposes is also crucial for the implementation of *automatic* capture of provenance metadata by workflow systems. Since the roles of provenance may vary between scientific communities, as well as the metadata that is required to address them, it is unlikely that there exists a single standard for provenance that is both detailed enough to address domain-specific requirements and yet still applicable across a broad range of scientific disciplines.

Renske: Append CWLProv paragraph to wf-centric RO or separate paragraphs?

To address this problem, we describe a systematic *approach* to identify the required provenance metadata for any scientific domain, and demonstrate how developers can use this information to improve existing provenance solutions. Instead of analyzing the high-level characteristics of many workflows, we decided to study a single workflow in detail, hypothesizing that the insights we would gain could also benefit workflows in other disciplines. We chose one exemplary bioinformatics workflow with which we were deeply familiar and defined a set of *provenance questions* associated with five realistic use case scenarios. Summarizing the metadata necessary to answer the questions in a *provenance taxonomy*, we analyzed how this information is captured in the current CWLProv community standard, distinguishing between different qualities of representation. Finally, we used the insights gained from our analysis to propose two extensions to CWLProv: Firstly, a set of recommendations for workflow authors to consistently annotate their input data, reusing existing, widely adopted standards. Secondly, we propose an improved representation of this metadata in CWLProv, to make it accessible for analysis with SPARQL [10], a query language for RDF.

Overall, our contribution is fourfold:

i. Description of a systematic approach to identify the required provenance metadata for any workflow, summarized into a provenance taxonomy.
ii. Qualitative analysis of the CWLProv community standard with respect to this provenance taxonomy.
iii. Design of an annotation scheme for workflow authors to consistently express required metadata as annotations to their input data.
iv. Extension of the CWLProv RDF provenance graph to make more provenance metadata accessible to analysis with SPARQL.

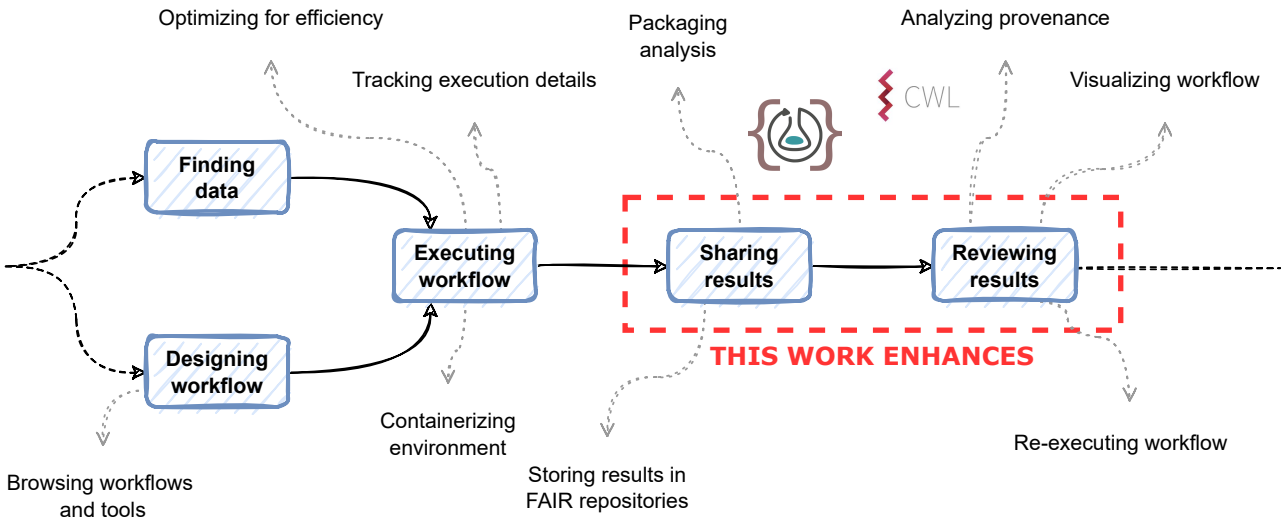

Figure 1. A vision of an ecosystem of workflow-related resources and its benefits for the reproducibility of computational processes. In this vision, workflows and data are stored in FAIR repositories, are executed in containerized environments by workflow systems which leverage efficient job-scheduling technologies and track details of the computation. After publication, the results of the workflow can be analyzed, and workflows can optionally be re-executed as the start of new projects which build upon the research. Fundamental to the success of this approach is the interoperability of the components of the ecosystem. In this work, we focus on workflow-centric Research Objects as a common format for the sharing and analysis of computational results.

Renske: The paper should address at least 4 audiences:
1. **CWL workflow authors** ('How do I make my workflow reproducible?')
2. **CWL developers** ('How to implement CWLProv in my workflow system?' 'Can I partly automate the provenance collection to lift the burden for workflow authors?')
3. **Provenance experts** ('But is this the proper way to represent provenance?' –> We want to tell them that a good provenance standard requires communication with practitioners: people with real-life workflows, versionless databases and annoying provenance issues. Standard must be useful/implementable in practice.)
4. **General FAIR/reproducibility/... enthusiasts**: Do not use CWL/ROs but care about FAIR. May maintain databases. ('How should I design the API for my database?' 'Should I send metadata during data download, and in which format?')
5. **Bioinformaticians** : the vast majority of workflows that are currently written do go nowhere near automated workflow, so we want to tell them to not rely on workflow systems to capture all the details, and they should start to think about that. Table 2 helps to think about all the things you should capture and worry about. Raise awareness. Figure 1 is the most important for them! Think about the entire process.

Results

Because metadata captured in ROs should be able to fulfill provenance requirements of real workflows, identification of these requirements is crucial to understand which metadata should be collected during execution, either by the workflow system or via manual annotations. However, due to the large variety of scientific communities and workflows, it is unlikely that a universal standard for provenance exists which is still sufficiently detailed to capture domain-specific elements.

Instead, we restricted ourselves to the detailed analysis of a single exemplary bioinformatics workflow, and formulated specific *provenance questions* grounded in a set of realistic use case scenarios. This imagination exercise gave us much insight into a set of important provenance metadata, and, summarizing it into a *provenance taxonomy*, we studied its representation in the current CWLProv community standard. Our analysis revealed that some elements of the taxonomy were insufficiently described, and that others were dependent on manual annotations of the input data and the workflow descriptions itself (for which the CWL standards were insufficiently clear); therefore our final step was to design extensions to CWLProv and guidance on manual annotations to address the gaps we had identified.

The rest of this section is organized as follows. Firstly, we describe the design and implementation of our exemplary workflow in CWL. Secondly, we define five realistic use case scenarios for ROs associated with the workflow. Subsequently, we synthesize a provenance taxonomy, based on a list of provenance questions grounded in each of the RO use case scenarios. Next, we describe the approach and results of our qualitative analysis of the current version of CWLProv (v0.6.0). Finally, we propose the design of two extensions to CWLProv: a scheme for annotation of workflow input data, and an extension of the provenance graph to enrich the RDF representation of the workflow execution.

Design and implementation of an exemplary bioinformatics workflow

As the basis of our provenance taxonomy, we used a type of workflow which is common in the field of bioinformatics: training a deep learning model to predict particular protein characteristics based on information encoded in the DNA. More specifically, the model in our example predicts, for a given protein, which amino acids constitute *epitopes*, i.e. binding sites for antibodies. Understanding antibody binding is important for e.g. vaccine design, but expensive and impractical to study experimentally because of the large number of possible protein-antibody combinations. However, the three-dimensional protein structure ultimately arises from the *protein sequence*, which has been determined for a far greater number of proteins and in principle can be used to predict structural properties which are difficult to determine in the lab.

Based on our experience, the workflow represents many of the problems which commonly arise in reproducing computational processes. Firstly, the model is an adaptation of a previously published model for predicting general protein binding characteristics (the protein-protein interface) [11]. Besides the main task, it also pre-

Renske: Nearly there. Last sentence has redundancies.

Renske: Combines **Data Description** + **Analyses** + **Methods**; https://academic.oup.com/gigascience/pages/research

Renske: Which "its" was studied?

Renske: Cite cwltool / cwlprov-py latest version or all versions?

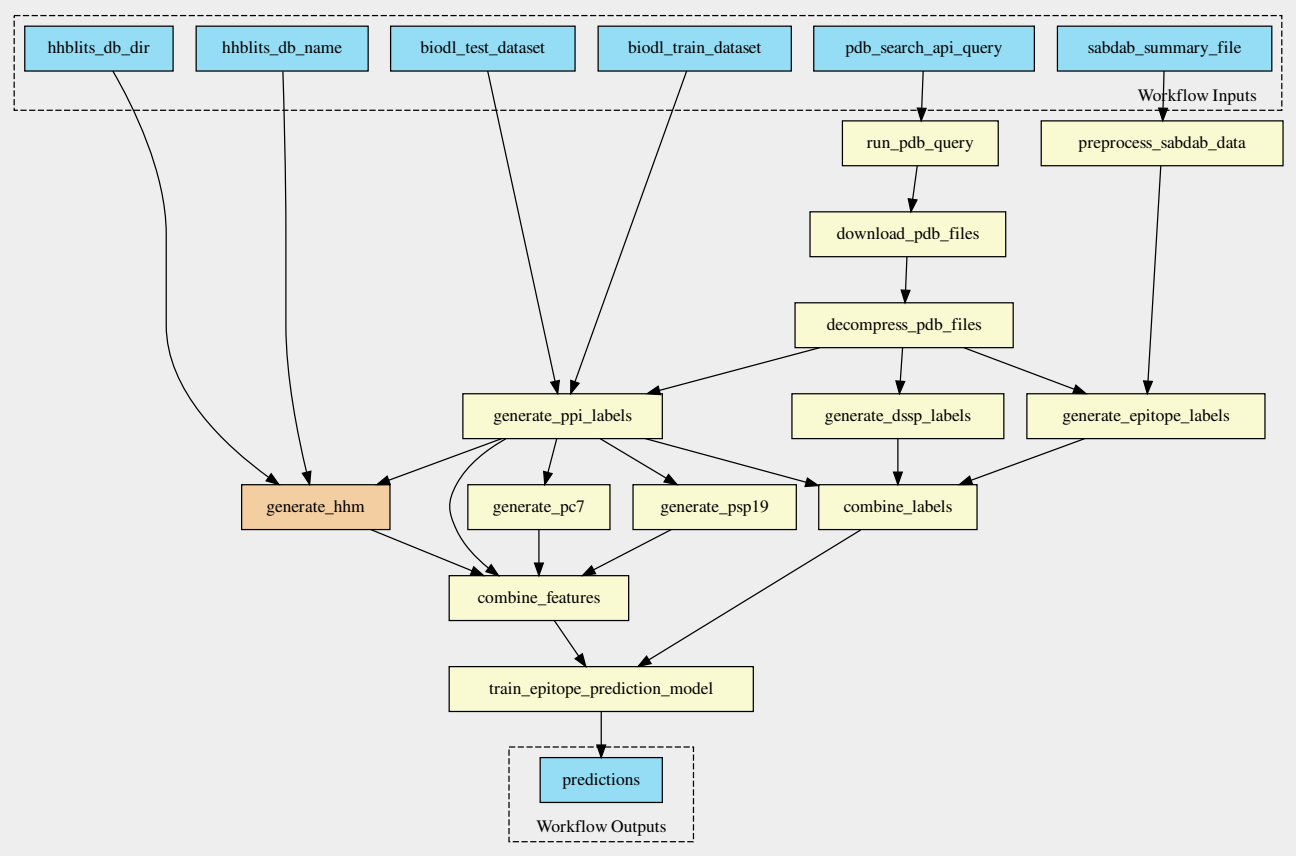

Figure 2. CWL implementation of the workflow we used as an example. Based on data retrieved from multiple FAIR and non-FAIR resources, the workflow computes a set of input features and labels. These are subsequently used to train a deep learning model which predicts the amino acids in a protein which are likely to be *epitopes*, i.e. binding sites for antibodies. Based on our experience, this workflow is exemplary for many problems which commonly arise when reproducing computational results. Nodes represent the steps, edges signify data flow between the steps. Yellow nodes indicate steps which run *CommandLineTools*, orange nodes represent steps which control nested *Workflows*, blue nodes signify workflow input and output parameters.

dicts several related structural properties (e.g. the fraction of each amino acid which is exposed to the outside of the protein). These labels are derived from experimentally resolved protein structures retrieved from large databases, which are continually updated and are unlikely to return the same results when the same query is issued at a later point in time. Secondly, the training features and labels are computed using a variety of tools with specific versions, dependencies, and input parameter settings. Moreover, multiple datasets are used as input, which requires mapping between different types of protein identifiers. Finally, workflow development usually involves many design choices, and many combinations of workflow steps and input configurations may be tested before the optimal configuration is found.

In our CWL implementation of the workflow, we aimed to optimize reproducibility. For example, we included database querying and data retrieval in the workflow and avoided the use of external identifier mapping tools, which are not likely to return the same output in the future. In addition, we executed most steps in software containers, and added structured annotations explaining the meaning of the workflow components and the data used as input. A detailed account of the workflow implementation can be found in the Supplementary Material.

Renske: Explicitly reference figure 2 in the text

Definition of RO use case scenarios

In this section, we describe the use cases of ROs associated with our example workflow coupled with the practitioners for which they are relevant. We consider five use cases, reflecting different stages in the workflow's lifetime. Some of these are applicable to workflows in most disciplines (e.g. U1), while at least one (U5) is specific to our exemplary workflow.

Renske: It is not really the DNA, but the protein sequence, but maybe this is less easy to understand.

U1 Workflow development. Most relevant for: **Workflow author**. During this stage, the workflow design is not fully established. Different input datasets and configurations are tested. Steps may be added or removed based on the output of previous steps. ROs of multiple workflow runs may be used to compare different designs and configuration settings with each other.

U2 Publishing the workflow. Most relevant for: **Workflow author**. At this stage, the workflow is ready for publication, and the workflow author has to explain the methodology and rationale of the research in a scientific article, or write documentation for the workflow. The RO may serve as a guide during writing, comprising a record of used data and tools, contributions of collaborators (facilitating credit and attribution), and choices which were made during workflow development.

U3 Understanding the workflow. Most relevant for: **Reader of the article describing the research**. An RO has been published in companionship with the article. The metadata contained in the RO serves as a bridge between the methods section of the article and the workflow itself. In addition, the metadata may connect the conclusions in the article to the results that support them.

U4 Reproducing the workflow. Most relevant for: **Third party continuing the research**. After reading the article and examining the RO to understand the research, this person may use the RO as a guide to reproduce the analysis first, before modifying or extending it for their own purposes.

U5 Model workflow execution as a service. Most relevant for: **User of the trained model as a web service**. In this stage, the trained model has been made available as a web service, which can

be used to predict epitopes for sequences for which the structure is unknown. An RO is the standard output format of the web-based tool.

Renske: explicitly where in the supplemental materials? with link

Synthesis of a taxonomy of provenance

Based on the use cases described in the previous section, we synthesized a list of *provenance questions*, included in the Supplementary Material. Similar to the use case scenarios, we found that some questions are likely also relevant in most other domains, while others are highly specific for our workflow. Subsequently, we identified the metadata required to answer the questions and summarized this into a 6-component taxonomy of provenance types which should be represented in the provenance of the workflow execution.

Renske: "Stages"? then use also in U3 & U4, or nowhere. Also "stages" feel a bit linear.

T1 **Scientific context**. Explanation of the choices which were made in the design of the workflow and parameter values.
T2 **Data**. Input and (intermediate) output data. To explain meaning and context of the data as well as describe data not included in RO for storage, security, or privacy reasons.
T3 **Software**. The tools directly orchestrated by the workflow, and their dependencies. Facilitates reproduction of workflow at a later time or helps finding an alternative tool in the case of code rot.
T4 **Workflow**. The (CWL) workflow and tool descriptions. Workflows are software and can be reused when published in a workflow repository.
T5 **Computational environment**. Metadata about the system on which the workflow was executed, comprising both software and hardware. Necessary for reproduction of the workflow execution.
T6 **Execution details**. Additional information about the workflow execution itself.

Each component of the taxonomy consists of several subcomponents, which are outlined in Table 1.

Qualitative analysis of the CWLProv community standard

In the previous section, we defined a set of metadata that should be included in provenance in order to meet specific use cases of ROs associated with an exemplary bioinformatics workflow. It is however also important to consider *how* the metadata is included in the RO. In this section, we describe a qualitative analysis of the current CWLProv community standard (v0.6.0), in order to decide how to extend the specification to meet the requirements associated with real-life workflows. We evaluated the representation of each component of the provenance taxonomy defined in the previous section, distinguishing between different qualities of representation.

Our analysis was based on RO Bundles generated with the CWL reference implementation *cwltool* . For each component of the provenance taxonomy, we evaluated the presence of the component in three levels of representation: structured representation in RDF (provenance graph and *manifest.json*), structured annotations in CWL-specific documents (*packed.cwl*, *primary-job.json*, and *primary-output.json*), and unstructured representation (e.g., the execution log).

Renske: explicitly where in the supplemental materials? with link

We also considered the amount of effort that is required to add the metadata: manual annotations require much more input from the workflow author than metadata that is automatically extracted by the workflow engine.

We considered a component *fully represented* if it is included in the document by default, or if the CWL Standards provide clear guidelines for its manual annotation in a workflow or input object document. In contrast, a provenance subtype was *partially represented* if only a subset of the metadata was included in the RO, or if there was ambiguity about how to represent this information in the workflow or parameter file.

Table 2 summarizes the results of the analysis. Of all 6 provenance elements, **execution details** (**T6**) are best represented in CWLProv. The provenance graph contains start and end timestamps at both workflow and step granularity(**EX1**). The workflow engine is described with name and version (**EX3**), and linked to the human agent who initiated the workflow run (**EX4**). Apart from maximum memory used during a container execution, the RO contains no reference to used resources (**EX2**).

The **workflow** (**T4**) is also represented in CWLProv. The full workflow is contained in *packed.cwl*, including any structured annotations made by the workflow author (**WF1**). In contrast, the RDF description references only the workflow and its steps. Annotations are not propagated to the provenance graph, and steps are not linked to the underlying *CommandLineTool* or nested *Workflow* descriptions. The workflow and step execution records mention the input parameters (**WF2**) and link them to their values, but they are not further explained with metadata in the provenance graph. Software and hardware requirements (**WF3**) can be found in *packed.cwl* if they were specified by the workflow author.

The **computational environment** (**T5**) is very sparsely described. Only when steps are executed in a software container, the container image (**ENV3**) is represented in RDF. However, the description is restricted to its name and tag, which is not stable and does not convey any information about its contents. If an image was built from a Dockerfile, the Dockerfile is included in *packed.cwl*, but there is no guarantee that the same image (with the same versions) will be built when re-executing the workflow. Other characteristics of the computational system (**ENV1**, **ENV2**) are completely absent from the provenance record.

Although the **software** (**T3**) orchestrated by the workflow may be part of *SoftwareRequirements* and therefore included in *packed.cwl*, there is no guarantee that the specified versions are identical to those installed on the system which executed the workflow. If workflow authors add structured annotations about the software to the *CommandLineTool* document, these are also contained in *packed.cwl*. None of these annotations are represented in RDF.

In contrast to software, **data entities** (**T2**) are represented in RDF, and linked to the workflow and step parameters for which they were values (**D4**). Each file is a separate entity in the provenance graph, and when they were part of an *Array* or *Directory*, this association is also described. Files are annotated in the provenance graph with filename and checksum, and *manifest.json* contains the creation date of files generated during workflow execution (**D2**). However, structured annotations of input data (**D1**, **D2**, **D3**) are restricted to *primary-job.json*.

The **scientific context** (**T1**) can be partially represented by adding manual annotations in the workflow (via *intent* and *doc* fields, **SC1**) and input file (via custom annotations, **SC2**) and will be present in *packed.cwl* and *primary-job.json*. There are currently no standards for adding annotations specific for the execution (hence **SC3** is missing).

Summarizing, we made three main observations about the CWLProv (v0.6.0) community standard. Firstly, the RDF graph is incomplete, limiting the use of SPARQL queries for analysis of the provenance. Secondly, although some of the missing metadata can be retrieved from CWL-specific documents, their presence is heavily dependent on manual annotations. The CWL Standards (v1.2) do not give detailed guidelines for the representation of annotations for input data, which is likely to increase the barrier for workflow authors to add these annotations to their workflows, and invites inconsistency when they do. Finally, there are some elements related to the execution which are missing from RDF and impractical to add as manual annotations. As a result, details about the computational environment are almost completely absent from the provenance record, compromising computational reproducibility of the workflow.

Table 1. Overview of provenance taxonomy. Integrated with already accepted principles and guidelines.

Renske: 1. link to provenance questions. 2. What should go in Refs? Only officially defined standards like FAIR principles/data citation principles or also random publications which mention that including something may be necessary?

Type	Subtype	Name	Metadata	Refs
T1	SC1	Workflow design	Annotations on the design of the workflow and its components. Purpose of the workflow, why steps were included or excluded, the meaning of particular input parameters, etc.	[1, 12, 13, 14]
	SC2	Entity annotations	The meaning of individual input and output data entities. Why were they chosen? How are the results interpreted?	
	SC3	Workflow execution annotations	Annotations about a set of parameters in a particular workflow run. Allows to distinguish between the ROs of multiple workflow runs.	
T2	D1	Data identification	Persistent identifier (PID), version, name and description of the dataset. Preferred citation of the data. *When the data is not FAIR:* URL and download date as an alternative for PID and version. *When the dataset is a subset of a larger collection (e.g. a database):* PID of database, database version and download date, and the query or filtering strategy which produced the dataset.	[15]
	D2	File characteristics	Filename, format, creation and last modification timestamps, size, and checksum.	
	D3	Data access	URL to a downloadable form of the data. License.	
	D4	Parameter mapping	The workflow and step parameters for which the data is an input or output.	
T3	SW1	Software identification	PID, name, version, release date and description of the software. Preferred citation. *When the software is not FAIR:* URL of repository, download date and/or git commit hash as substitute for PID, version or release date.	[16]
	SW2	Software documentation	URL of documentation or other metadata which is important to make informed use of the tool. URL to repository with source code of the software.	
	SW3	Software access	URL to downloadable, executable form of the software. License.	
T4	WF1	General software metadata	At workflow and step level, according to T3.	[17]
	WF2	Workflow parameters	Type, format, and description, at workflow and step-level.	
	WF3	Workflow requirements	Software and hardware resources which are required to execute the workflow or workflow steps.	
T5	ENV1	Software environment	Software (dependencies), operating system. Dependencies could comprise all installed software (might contain much redundant information if a step was not executed in a software container), or the dependencies of the software which is run (which may be difficult to identify). Should follow the requirements as described in T3.	[1]
	ENV2	Hardware environment	Available RAM, storage, number and type of CPUs and GPUs. Network access.	
	ENV3	Container image	Image name, tag and digest (because names and tags are not stable). Additional metadata (extracted from image labels), contents of Dockerfile (if built from Dockerfile), and general requirements for software as described in SW1.	
T6	EX1	Execution timestamps	When the workflow was executed, at step-granularity. The timestamps can be helpful when files were downloaded during the execution, especially from a database which does not have clear versions. In addition, the duration of the execution may be important during workflow development (test different settings) and when reproducing the workflow.	
	EX2	Consumed resources	The resources used during execution, at step-granularity. This is different from what was described in WF3, because there we only described what was available, not what was actually used.	
	EX3	Workflow engine	Software, therefore with same metadata as general software entities (T3).	
	EX4	Human agent	At a minimum, a PID such as ORCID should be included, or name and email of the person who ran the workflow. These details may be important for attribution (U2), and can also be used by third parties to ask further questions about the research.	

Proposed scheme for the annotation of input data

In the previous sections, we identified a set of useful provenance metadata given 5 realistic use case scenarios, summarized this into a taxonomy, and used this to perform a qualitative analysis of the current CWLProv community standard. From this analysis, we concluded that although manual annotations can be added to the workflow and parameter files to enrich the provenance, the CWL

Table 2. Summarized results of CWLProv 0.6.0 analysis. The table shows for each taxonomy component whether it is fully (■) or partially represented (□). Components which are not represented in any of the documents are marked with an asterisk (∗). Representations which are only included when this metadata is manually supplied are indicated with parentheses. We distinguish between description in RDF, CWL-specific documents (packed.cwl, primary-job.json, and primary-output.json), and unstructured representation (e.g. execution log).

Type	Subtype	Name	structured (RDF)	CWL-specific	unstructured
T1	SC1	Workflow design		(■)	
	SC2	Entity annotations		(□)	
	SC3	Workflow execution annotations ∗			
T2	D1	Data identification		(□)	
	D2	File characteristics	□	(□)	□
	D3	Data access		(□)	
	D4	Parameter mapping	■	■	□
T3	SW1	Software identification		(□)	
	SW2	Software documentation		(□)	
	SW3	Software access		(□)	
T4	WF1	Workflow software metadata	□	(■)	
	WF2	Workflow parameters	□	(■)	
	WF3	Workflow requirements		(■)	
T5	ENV1	Software environment ∗			
	ENV2	Hardware environment ∗			
	ENV3	Container image	□	□	□
T6	EX1	Execution timestamps	■		
	EX2	Consumed resources			□
	EX3	Workflow engine	□		
	EX4	Human agent	■		■

Standards do not formally specify how this information should be included. In this section, we aim to standardize the annotation of input data, proposing a scheme which enables authors to describe individual inputs as well as a complete workflow run (i.e., the combination of a particular workflow design and its configuration settings, **SC3**).

Rather than designing our own ontology, the core vocabulary of our annotation scheme reuses the Bioschemas [18] *Dataset* profile (v1.0), an extension of Schema.org [19] for life sciences. The terms listed in Table 3 allow authors to express generic characteristics of their input datasets, such as the identifier, version, and license. In addition, to convey information about the *meaning* of the data, we recommend the use of other, domain-specific ontologies, e.g. EDAM [20].

In many cases, input datasets may be the result of a series of operations prior to workflow execution, from retrieval from a database to filtering and cleaning operations. We advise to encode this history as a sequence of Schema.org *Actions*, each associated with their own instruments, queries, and times (Table 4).

Figure 3 shows an example of how to apply the annotation scheme in practice. This is a parameter file with two workflow input values. The first is a standalone dataset with its own identifier and version. The second dataset is the result of a database query and subsequent filtering. The metadata is separated from the rest of the input parameters by a dedicated *cwlprov:prov* field. Finally, a description of the entire workflow run is given in the root of the document.

The details of the design, as well as more extensive examples of how to apply the scheme in practice, are provided in the Supplementary Material.

Renske: mention "–provenance" or otherwise link to details on how this was done.

Proposed extension of the RDF provenance graph

In our analysis of CWLProv, described in the previous sections, we observed that much of the metadata was absent from the RDF representation of the execution. In this section, we describe the design and implementation of an extension to the provenance graph, in order to make more of the analysis accessible to exploration with SPARQL queries.

Table 3. Schema.org terms to use to express the metadata elements described in **T2**. Taken from Bioschemas Dataset profile v1.0.

Schema.org	T2 element	Expected type
identifier	PID	URL
version	version	Number, Text
name	name	Text
description	description	Text
citation	citation	CreativeWork
includedInDataCatalog	database	DataCatalog
dateCreated	download date	Date, DateTime
dateModified	modification date	Date, DateTime
distribution	URL to data	DataDownload
license	license	URL

Table 4. Schema.org terms to represent the history of data inputs.

Schema.org	Expected type	Explanation
query	Text	Query used (only for SearchAction)
object	Thing	Database or initial dataset
result	Thing	Resulting dataset
instrument	Thing	Tool used, e.g. for filtering
endTime	DateTime, Time	Time of action
agent	Organization, Person	Who performed the action
description	Text	Description of the action

Primarily, we aimed to expand the description of the workflow (i.e. *prospective provenance*). Whereas the original design only mentioned the top-level workflow and its steps, the extended design incorporates all components for which the CWL Standards specify metadata fields (Table 5). In addition, provenance metadata supplied according to the annotation scheme described in the previous section is propagated to RDF. The details of the design are described in Supplementary Materials.

Figure 3. Simplified example of our annotation scheme for workflow inputs. The first input ($standalone_{dataset}$) is a dataset stored in a FAIR repository, with an identifier and version. The second input ($filtered_{pdb_{dataset}}$) is the result of a database search (pdb_{search}) and subsequent filtering ($filtering_{action}$).

```
cwlprov:prov:
  pdb_search:
    s:additionalType: s:SearchAction
    s:query: "All proteins with at least 2 chains deposited between 2010 and 2022"
    s:object:
      s:identifier: https://bio.tools/pdb
    s:result: pdb_search_result
    s:endTime: 2022-08-01

  filtering_action:
    s:additionalType: s:Action
    s:object: pdb_search_result
    s:instrument:
      s:identifier: https://bio.tools/pisces
    s:result: filtered_pdb_dataset

filtered_pdb_dataset:
  class: Directory
  location: path://path/to/directory/

standalone_dataset:
  class: File
  location: path://path/to/file1.pdb
  format: edam:format_1476 # pdb
  s:identifier: https://doi.org/10.2210/pdb6nzn/pdb
  s:version: "1.4"
  s:description: "Amyloid fibril structure of glucagon in pdb format."

s:description: "Example workflow run with 2 inputs."

$namespaces:
  s: "https://schema.org/"
  edam: "http://edamontology.org/"

$schemas:
- https://schema.org/version/latest/schemaorg-current-https.rdf
- https://edamontology.org/EDAM_1.25.owl
```

Table 5. Metadata fields in CWL Standards v1.2. *format* is only allowed for parameters of type File or File array.

Workflow component	label	doc	intent	format
Workflow	·	·	·	
WorkflowStep	·	·		
CommandLineTool	·	·	·	
ExpressionTool			·	
WorkflowInputParameter	·	·		·
WorkflowOutputParameter	·	·		·
WorkflowStepInput	·			
WorkflowStepOutput				
CommandInputParameter	·	·		·
CommandOutputParameter	·	·		·

We realized the extended design in the CWL reference implementation *cwltool*. Repeating the analysis of CWLProv, we found that the extended RDF graph is notably richer in provenance metadata (Table 6). Although still dependent on manual input from the workflow author, **Scientific context (T1)** is now fully represented in RDF, as well as **Data (T2)** and **Workflow (T4)**. Future work should focus on the representation of workflow resource requirements, as well as computational environment. A list of SPARQL queries can be found in the Supplementary Material.

Renske: explicitly where in the supplemental materials? with link

Discussion

To facilitate reuse of computational research, there is an urgent need for a common, machine-accessible format for sharing the results of computational analyses. CWLProv [4] is an RO specification that encapsulates both data and provenance metadata associated with workflow executions. However, the wide choice in metadata standards and ontologies can make it difficult for practitioners and developers to *prioritize* a set of useful metadata that should be collected during workflow execution. Here we describe an approach to define such a set of metadata by focusing on relevant provenance questions grounded in a limited set of use cases for one exemplary workflow, instead of a high-level analysis of many workflows in a particular discipline.

We summarize this metadata into a *provenance taxonomy* and perform a systematic analysis of the CWLProv community standard, distinguishing between three different levels of representation. From this analysis, we identify three areas of improvement for CWLProv: to describe the computational environment on which the workflow is executed, propagate annotations in the workflow and input parameter file to the RDF provenance record, and pro-

Renske: TODO (Michael)

Renske: TODO (Renske)

vide CWL practitioners with detailed guidelines for the annotation of their input data. To improve CWLProv, we propose two extensions. Firstly, we reuse the Bioschemas [18] *Dataset* profile to design a scheme for the annotation of input data. Secondly, we extend the RDF provenance graph to include all workflow components for which CWL-specific metadata fields have been defined (Table 5), which is compatible with the proposed annotation scheme.

The approach described here makes adding provenance to workflows *actionable*. In the first place, creating a list of practitioner-centered provenance questions helps communities prioritize the provenance metadata they should share with their data. Importantly, we do not intend to define a new metadata standard, but rather help communities apply existing standards, selecting those that are most relevant for the purposes for which provenance is used in their domain.

In addition, our approach is also highly informative for the development and improvement of RO specifications. We demonstrated this by designing two extensions to CWLProv to address the gaps we identified in our systematic analysis of the specification. Moreover, our taxonomy was also used in the design of the CWLProv successor, the Workflow Run RO-Crate profile[1], as part of the RO-Crate project [21].

The first of our extensions is an annotation scheme for input data. In the current version of CWLProv and its implementation in *cwltool* [5], provenance of input data is only captured as structured annotations in the input parameter file. To promote consistency, we propose detailed guidelines for the annotation of single inputs as well as complete workflow runs, which are lacking in the current CWL Standards (v1.2)[2] [2] . To optimize the tradeoff between domain-neutrality and expressiveness, the core of our scheme is based on Schema.org [19], but we support the use of additional ontologies for domain-specific concepts. If necessary, individual communities can define specializations to our scheme that more specifically address their provenance needs.

Although the annotation scheme *conceptually* provides practitioners with a means to add metadata to their data, our ultimate goal is to capture workflow execution provenance automatically (Fig. 1). Firstly, many useful metadata is probably already accessible to workflow systems, especially if data retrieval is part of workflow execution. In addition, it is possible to lower the barrier for manual annotation (e.g. to capture the *why* of the analysis, T1) by providing prompts in the workflow system interface, subsequently converting them to semantic annotations. To promote interoperability, future work should focus on defining a common, machine-accessible format in which databases can supply their data and metadata. We believe that an RO specification (possibly an RO-Crate profile) could be a suitable candidate for this purpose.

Finally, further research should focus on the development of tools which facilitate provenance analysis, for example through standardized queries (the provenance questions and taxonomy can act here as a guideline), which would remove the need for scientists to learn SPARQL before they can analyze the RO.

Finally, our extension of the RDF provenance graph enables more comprehensive querying with SPARQL [10]. Nevertheless, computational environment (T5), an important factor in workflow executions, is still insufficiently represented for the use cases we considered here. This was predominantly caused by the lack of a suitable standard for the representation of this metadata. Future work should focus on the development of such a standard, preferably in a working group including a wide array of stakeholders, in order to be broadly adopted.

In conclusion, the complexity of present-day computational analyses necessitates the development of automatic provenance tracking solutions. The results of this work can guide the development of these solutions, since they help prioritize which metadata is most useful, based on the purposes which are relevant from a bioinformatics practioner's perspective.

Table 6. RDF provenance graph after extension. The table shows for each taxonomy component whether it is fully (black) or partially represented (white), highlighting elements with improved representation in the extended design (★). Components which are not represented in any of the documents are marked with an asterisk (∗). Representations which are only included when this metadata is manually supplied are indicated with parentheses.

Type	Subtype	Name	structured (RDF)
T1	SC1	Workflow design	(★)
	SC2	Entity annotations	(★)
	SC3	Workflow execution annotations	(★)
T2	D1	Data identification	(★)
	D2	File characteristics	□
	D3	Data access	(★)
	D4	Parameter mapping	■
T3	SW1	Software identification	
	SW2	Software documentation	
	SW3	Software access	
T4	WF1	Workflow software metadata	(★)
	WF2	Workflow parameters	(★)
	WF3	Workflow requirements	
T5	ENV1	Software environment ∗	
	ENV2	Hardware environment ∗	
	ENV3	Container image	□
T6	EX1	Execution timestamps	■
	EX2	Consumed resources	
	EX3	Workflow engine	□
	EX4	Human agent	■

1 https://www.researchobject.org/workflow-run-crate/
2 https://www.commonwl.org/v1.2/

Availability of source code and requirements (optional, if code is present)

...

Availability of supporting data and materials

...

Declarations

List of abbreviations

- CWL: Common Workflow Language
- PID: Persistent Identifier
- RO: Research Object
- RDF: Resource Description Framework
- SPARQL: SPARQL Protocol and RDF Query Language

Consent for publication

Not applicable.

Competing Interests

The authors declare they have no competing interests.

Renske: Combines **Discussion + Potential implications** sections; https://academic.oup.com/gigascience//pages/research

Funding

Renske: @Michael I don't know.

Author's Contributions

- RdW: formal analysis, investigation, software, visualization, writing - original draft.
- SA: conceptualization, funding acquisition, methodology, writing - review and editing.
- AI: conceptualization, funding acquisition, methodology, supervision, writing - review and editing.
- MRC: conceptualization, methodology, software, supervision, writing - review and editing.

Renske: Is there a DOI for CWL standards v1.2?

Acknowledgements

BAZIS compute cluster

References

1. Committee on Reproducibility and Replicability in Science, Board on Behavioral, Cognitive, and Sensory Sciences, Committee on National Statistics, Division of Behavioral and Social Sciences and Education, Nuclear and Radiation Studies Board, Division on Earth and Life Studies, et al. Reproducibility and Replicability in Science. Washington, D.C.: National Academies Press; 2019.
2. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanić N, et al. Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language. Communications of the ACM 2022 Jun;65(6):54–63.
3. Belhajjame K, Corcho O, Garijo D, Zhao J, Missier P, Newman D, et al. Workflow-Centric Research Objects: First Class Citizens in Scholarly Discourse. In: 9 Th Extended Semantic Web Conference Hersonissos Hersonissos, Crete (Greece); 2012. p. 1–12.
4. Khan FZ, Soiland-Reyes S, Sinnott RO, Lonie A, Goble C, Crusoe MR. Sharing Interoperable Workflow Provenance: A Review of Best Practices and Their Practical Application in CWLProv. GigaScience 2019 Nov;8(11):giz095.
5. Crusoe MR, Khan FZ, Amstutz P, Soiland-Reyes S, Singh M, Kumar K, et al., Common-Workflow-Language/Cwltool: 3.1.20230513155734; 2023. Zenodo.
6. Cyganiak R, Wood D, Lanthaler M, Klyne G, Carroll JJ, McBride B, RDF 1.1 Concepts and Abstract Syntax; 2014. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.
7. Moreau L, Missier P, Belhajjame K, B'Far R, Cheney J, Coppens i, et al., PROV-DM: The PROV Data Model; 2013. https://www.w3.org/TR/2013/REC-prov-dm-20130430/.
8. Soiland-Reyes S, Crusoe MR, Khan FZ, Leinweber K, Bot S, Common-Workflow-Language/Cwlprov-Py: 0.1.2; 2022. Zenodo.
9. Missier P. The Lifecycle of Provenance Metadata and Its Associated Challenges and Opportunities. In: Lemieux VL, editor. Building Trust in Information Cham: Springer International Publishing; 2016.p. 127–137.
10. The W3C SPARQL Working Group, SPARQL 1.1 Overview; 2013. http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.
11. Capel H, Feenstra KA, Abeln S. Multi-Task Learning to Leverage Partially Annotated Data for PPI Interface Prediction. Scientific Reports 2022 Jun;12(1):10487.
12. Belhajjame K, Zhao J, Garijo D, Hettne K, Palma R, Corcho Ó, et al. The Research Object Suite of Ontologies: Sharing and Exchanging Research Data and Methods on the Open Web. arXiv:14014307 [cs] 2014 Feb;.
13. Gryk MR, Ludäscher B. Workflows and Provenance: Toward Information Science Solutions for the Natural Sciences. Library Trends 2017;65(4):555–562.
14. Stodden V, McNutt M, Bailey DH, Deelman E, Gil Y, Hanson B, et al. Enhancing Reproducibility for Computational Methods. Science 2016 Dec;354(6317):1240–1241.
15. Data Citation Synthesis Group. Joint Declaration of Data Citation Principles. San Diego CA: FORCE11 2014;.
16. Smith AM, Katz DS, Niemeyer KE, FORCE11 Software Citation Working Group. Software Citation Principles. PeerJ Computer Science 2016 Sep;2:e86.
17. Goble C, Cohen-Boulakia S, Soiland-Reyes S, Garijo D, Gil Y, Crusoe MR, et al. FAIR Computational Workflows. Data Intelligence 2020 Jan;2(1-2):108–121.
18. Michel F, The Bioschemas Community. Bioschemas & Schema.Org: A Lightweight Semantic Layer for Life Sciences Websites. Biodiversity Information Science and Standards 2018 May;2:e25836.
19. Guha RV, Brickley D, Macbeth S. Big Data Makes Common Schemas Even More Necessary 2015;p. 28.
20. Ison J, Kalas M, Jonassen I, Bolser D, Uludag M, McWilliam H, et al. EDAM: An Ontology of Bioinformatics Operations, Types of Data and Identifiers, Topics and Formats. Bioinformatics 2013 May;29(10):1325–1332.
21. Soiland-Reyes S, Sefton P, Crosas M, Castro LJ, Coppens F, Fernández JM, et al. Packaging Research Artefacts with RO-Crate. Data Science 2022 Jan;p. 1–42.
22. Xu G, Wang Q, Ma J. OPUS-TASS: A Protein Backbone Torsion Angles and Secondary Structure Predictor Based on Ensemble Neural Networks. Bioinformatics 2020 Dec;36(20):5021–5026.
23. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, et al. The Protein Data Bank. Nucleic Acids Research 2000 Jan;28(1):235–242.
24. Dunbar J, Krawczyk K, Leem J, Baker T, Fuchs A, Georges G, et al. SAbDab: The Structural Antibody Database. Nucleic Acids Research 2014 Jan;42(D1):D1140–D1146.
25. Stringer B, de Ferrante H, Abeln S, Heringa J, Feenstra KA, Haydarlou R. PIPENN: Protein Interface Prediction from Sequence with an Ensemble of Neural Nets. Bioinformatics 2022 Apr;38(8):2111–2118.
26. Remmert M, Biegert A, Hauser A, Söding J. HHblits: Lightning-Fast Iterative Protein Sequence Searching by HMM-HMM Alignment. Nature Methods 2012 Feb;9(2):173–175.
27. Kabsch W, Sander C. Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. Biopolymers 1983 Dec;22(12):2577–2637.
28. The UniProt Consortium, Bateman A, Martin MJ, Orchard S, Magrane M, Agivetova R, et al. UniProt: The Universal Protein Knowledgebase in 2021. Nucleic Acids Research 2021 Jan;49(D1):D480–D489.
29. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific Containers for Mobility of Compute. PLOS ONE 2017 May;12(5):e0177459.
30. Gray AJG, Goble C, Jimenez RC. From Potato Salad to Protein Annotation 2017 Oct;p. 5.

Renske: Replaces **Methods** section. https://academic.oup.com/gigascience//pages/research

SUPPLEMENTARY

Use case workflow

Renske: Text in supplementary is for a large part identical to the thesis, not paraphrased

Renske: add intro section here

Requirements

The CWL implementation of our example workflow should meet two requirements, which we explain further in the subsequent sections:

WR1 Adhere to a specific method of epitope prediction which was conceptualized and provided to us by the workflow authors.
WR2 Address challenges which compromise the transparency and reproducibility of this workflow.

WR1: Adhere to a conceptual method

In this section, we explain **WR1**, providing a conceptual overview of the method which should be implemented in the CWL workflow we used as an exemplary workflow in this work.

The design of the epitope predictor, conceptualized by the workflow authors, is based on a model for predicting 'general' protein-protein interaction (PPI) interfaces [11]. In its turn, that model was derived from OPUS-TASS [22], a predictor for protein *structure*.

It addresses the lack of training data via a multi-task learning strategy, in which the model not only learns the label of interest (epitopes), but also related characteristics, such as solvent accessibility (the fraction of amino acid surface which is exposed to the aqueous environment). This allows the model to be trained on a larger set of structures, not restricted to antibody-antigen complexes, but also including structures with general PPI interfaces and other structural information.

The PPI predictor on which this method was based, was trained on OPUS-TASS reference data and did not include calculation of the input features or labels. In contrast, the OPUS-TASS source code did contain calculation of the input features, but the labels were reused from a reference dataset and not directly derived from protein structure. **However, because we wanted to maximize the transparency of our research, our workflow comprises the entire trajectory from protein structure and sequence to labels and features used for model training.**

Data sources

These are the (main) data sources used to calculate the input features and labels:

- **Protein Data Bank (PDB)** [23], a database of experimentally resolved structures of proteins and complexes.
- **Structural Antibody Database (SAbDab)** [24], a database of metadata for antibody-antigen complexes in the PDB (e.g., which part of the complex corresponds to the antibody and which to the antigen).
- **BioDL** [25], a dataset of protein sequences containing annotations for general PPI interactions, previously derived from structure.
- **HHBlits reference database.** Used by HHBlits [26] to compute some of the input features.

Structure-derived labels

For each residue in each protein sequence, the model predicts a number of properties.

- **Epitope**: a binary label indicating if a given residue is an epitope. Calculated from protein structure in combination with SAbDab metadata. Missing for proteins which are not included in SAbDab.
- **PPI**: a binary label indicating if a given residue is part of a general PPI interface. Extracted from BioDL.
- **Surface accessibility**: a label indicating how much of the amino acid is exposed to the surface. Calculated by DSSP [27] based on the protein structure.
- **Secondary structure**: 3 binary labels indicating the secondary structure of which a given residue is part (alpha helix, beta strand or loop). Calculated by DSSP based on the protein structure.

Sequence-derived input features

The model predicts epitope annotations based on three groups of sequence-derived features:

- **PC7**: 7 features which reflect amino-acid-specific physicochemical characteristics. Every amino acid type has fixed values for these features.
- **PSP19**: 19 binary features which reflect the presence of particular amino acid 'building blocks' (e.g. a benzene ring). Every amino acid type has fixed values for these features.
- **HHM**: 30 features derived from a sequence profile generated from alignment with highly similar sequences. Computed with HHBlits.

WR2: Address reproducibility challenges for this workflow

In this section, we explain **WR2** by presenting a number of characteristics of this workflow which may make its implementation challenging from a transparency and reproducibility perspective.

Workflow design.

Firstly, the method requires calculation of over 50 input features and at least 6 labels for every amino acid in each of the several thousand proteins in the training dataset, involving three data sources, at least two command-line tools and several Python scripts. This underlines the importance of describing this process as a workflow, in order to keep track of all the data flows.

Secondly, the design of the workflow is subject to extensive changes, as the workflow authors test different combinations of input features and labels in order to optimize performance as well as computational efficiency. In addition, they may need to include extra preprocessing steps to remove potential bias from the training set (e.g. due to overrepresentation of certain protein families in the PDB).

Software

The workflow steps each have their respective software dependencies, some of which are only compatible with particular versions of other software (e.g. *tensorflow*). Recording the versions of tools and dependencies is therefore very important for reproducibility.

Data

Retrieving and handling the data used as input for this workflow has its own challenges. Firstly, HHBlits can be used with different reference databases, which will influence the produced sequence profiles. These databases have versions, and are not stored in a FAIR manner.

Secondly, the dataset downloaded from PDB does not comprise the entire database, but a subset which is selected as the result of a query. Since new entries are added to PDB continually, it is not likely that running the same query at a later moment will result in the same dataset. Similarly, SAbDab also receives weekly updates [24].

Finally, the identifiers in the BioDL dataset correspond to particular *sequences*, whereas those in SAbDab and PDB represent *structures*. Therefore, we need to match the two types of identifiers with each other. However, external resources such as the UniProt mapping tool[3] [28] may not return the same mappings in the future.

CWL implementation of the workflow

In this section, we describe how we implemented the epitope prediction workflow in CWL, considering the requirements described in the previous sections. Figure 2 shows an overview of the CWL implementation of the workflow.

Implementation of the conceptual method (WR1).
To address the first requirement, the workflow starts by issuing a query to the PDB Search API[4] (*run_pdb_query*). This produces a list of PDB IDs which is used to download the protein structures from PDB (*download_pdb_files*), which are subsequently decompressed (*decompress_pdb_files*). From the protein structures, DSSP calculates surface accessibility and secondary structure (*generate_dssp_labels*), and an in-house Python script extracts epitope annotations (*generate_epitope_labels*) using a SAbDab summary file which has been preprocessed in an earlier step (*preprocess_sabdab_data*). Another Python script extracts PPI annotations from BioDL and performs identifier mapping (*generate_ppi_labels*). Three separate steps calculate input features for the protein sequences with PPI annotations (*generate_hhm*, *generate_pc7*, and *generate_psp19*). The input features and labels are subsequently combined in two steps (*combine_features*, *combine_labels*) and used to train the prediction model (*train_epitope_prediction_model*).

Consideration of workflow reproducibility (WR2).
Firstly, we aimed to automate the workflow as much as possible. For this reason, the PDB query and download steps are included in the workflow (with the query as one of the workflow inputs). The two workflow inputs constituting the BioDL dataset (*biodl_test_dataset* and *biodl_train_dataset*) are included as remote files and downloaded by *cwltool* during workflow execution. Because SAbDab is not programmatically accessible, *sabdab_summary_file* needs to be downloaded manually, but all further preprocessing steps are included of the workflow.

To avoid using external mapping tools between UniProt and PDB identifiers, we infer the relationships between the two types of IDs from the downloaded PDB files, which contain both types of identifiers.

To make the workflow design easy to adapt and modify with different combinations of input features and labels, we spread the computation of these features over separate steps. This is different from the original OPUS-TASS code, in which all input features are calculated by a single Python script.

To increase the portability of the computational environment, some steps are executed inside software containers. The workflow engine pulls Docker images from external repositories and converts them to Singularity [29] containers, the software which is installed on the Bazis HPC cluster on which we executed the workflow.

Emulation of workflow steps.

Renske: Some information may be irrelevant, other info needs to be rephrased.

Because this workflow is still in development and not all the scripts were ready, some of the steps are emulated. In this way, the steps produce output in the expected format, but do not necessarily contain biologically sensible information.

In summary, for three steps we used scripts provided by the workflow authors (*preprocess_sabdab_data*, *generate_epitope_labels*, and *generate_dssp_labels*). For other steps we reused or modified code from OPUS-TASS (*generate_pc7*, *generate_psp19*, and *generate_hhm*) or other sources (*run_pdb_query*, *download_pdb_files*). Finally, we wrote custom Python scripts for the remaining steps, of which two were partially (*combine_labels*) or completely emulated (*train_epitope_prediction_model*).

3 https://www.uniprot.org/id-mapping/
4 https://search.rcsb.org/index.html#search-api

Provenance questions

Renske: How to organize this section? Idea 1: leave it as it is. Idea 2: Make sideways table with 1 question per row, add columns for each use case (U1 - U5) and provenance subcomponent (SC1, SC2, SC3, D1, ...). Indicate for which use case and provenance component each question is applicable.

U1: Workflow development

T1 Context

i. What is the difference between workflow runs X and Y?

T2 Data

i. What is the influence of a different training set on performance?
ii. What is the influence of a different value for input parameter X on performance?

T3 Software

i. What is the influence of a different model architecture on performance?
ii.

T4 Workflow

i. What is the influence of removing model training feature X on performance?
ii. What is the influence of including filtering step X on performance?

T6 Execution

i. What is the influence of ... on training time?
ii. Which of workflow runs X and Y used more RAM?

U2: Publishing the workflow

T1 Context

i. Which of my colleagues contributed to the conceptualization of this research? What were their contributions?
ii. On which previous research was this research built (conceptually)?

T2 Data

i. Which were all the datasets I used as inputs and should be cited?
ii. Which cleaning operations were performed on workflow input X?
iii.

T3 Software

i. What are all the resources which contributed to this research (and should be cited)?
ii. Which software was used in this workflow?
iii. I wrote a CWL *CommandLineTool* description for existing software. How do I cite it?
iv. I reused a custom, unpublished script made by one of my collaborators. How do I give credit to the original authors?
v. I wrote a custom script based on existing code. How do I give credit to the original authors?

T4 Workflow

i. I reused an existing CWL CommandLineTool description. How do I give credit to the original authors?
ii. Which of my colleagues contributed to this workflow to whom I should give credit and/or propose as co-authors? What were their contributions?
iii. From which workflow(s) was this workflow derived?

T5 Environment

i. I reused a Dockerfile or Docker container which was written or built by someone else. How do I give credit to the original authors?

T6 Execution

i. When was dataset X downloaded from database Y?

U3: Understanding the workflow

T1 Context

i. What is the goal of this analysis? What was the hypothesis of this experiment?
ii. Why was this step included?
iii. Why was this combination of steps chosen?
iv. Why was this set of values chosen as inputs for this workflow execution?
v. Why were these input settings chosen?
vi. What is the interpretation of this (intermediate) output?

T2 Data

i. How was this figure from the paper generated?
ii. What was the performance of the model?
iii. Where can I find more information about BioDL dataset?
iv. Which SAbDab query was used to generate the summary file?
v. Which UniProt IDs are part of BioDL?
vi. Which query was issued to PDB?

T3 Software

i. Which related tasks were predicted by the model?
ii. What was model architecture?

T4 Workflow

i. How were UniProt IDs mapped to PDB IDs?
ii. Which input features were used for the model?
iii. What are the values of PSP19 for each amino acid type?

T6 Execution

i. Who can I contact for more information?

U4: Reproducing the workflow

T2 Data

i. What was the last modification date of this input file?
ii. Which HHBlits reference database was used? Which version?
iii. Where can I download HHBlits reference database? (It was not stored in the RO because of its large size.)

T3 Software

i. The CWL *CommandLineTool* description did not describe all parameters of this command-line program. Which values were used for the other parameters?

ii. Which version of HHBlits was used?
iii. Which format does the PDB batch download script need?

T4 Workflow

i. Does this step need network access?
ii. What are resource requirements as specified in the workflow description?
iii. How many CPUs are required for this step?
iv. How much memory is necessary for this step?

T5 Environment

i. Which Python version was used in this step?
ii. Which software (and their versions) was installed on the system?
iii. How much memory was used in the original run?
iv. When was PDB queried?
v. When was SAbDab queried?
vi. Original author used Dockerfile. Is the image I built on a rerun the same as the one in the original analysis?
vii. Which CPU/GPU was installed?
viii. I pulled a Docker image. Is this the same as which was used in the original analysis?

T6 Execution

i. How long does this step take?
ii. Who ran the original workflow and how can I contact them for more information?

U5: Model as a web service

T2 Data

i. What was the query which returned the set of proteins for which I made predictions with the model?
ii. Which protein sequences were in the training set?
iii. Which proteins had epitope annotations?

T4 Workflow

i. How should I cite this tool?

Annotation scheme

Requirements

Based on the results of the analysis described in , the input annotation scheme should meet the following requirements:

IR1 Represent the elements defined in **??** and **??**.
IR2 Describe input data of type *File*, *Directory* and *Array*.
IR3 Represent the history of processed input data (e.g filtering).
IR4 Represent the database query which produced a dataset.
IR5 Support extension with domain-specific vocabularies.
IR6 Represent information about a set of input parameters (**??**)

Design principles

The design of the input data annotation scheme was based on a set of underlying principles:

IP1 **Reuse of existing terms and ontologies.** Our scheme uses Schema.org, since this complies with the Bioschemas [30] initiative. Schema.org terms are also adopted by related efforts such as the RO-Crate specification [21].
IP2 **Extension of the CWL standards only when absolutely necessary.** We started from the latest CWL Standards specification (v1.2), and only where these did not support adding metadata we proposed an extension.
IP3 **Clear separation between input data and metadata.** This keeps the input object document relatively easy to understand.
IP4 **Simplicity.** The annotation scheme should be easy to understand and use for CWL workflow authors.

Annotations supported by CWL standards v1.2

Here, we explain the annotations that are supported by the latest release of the CWL standards (**IP2**). In the examples outlined below, we abbreviate *http://schema.org/* with the prefix *s:*.

Format
CWL Standards v1.2 support semantic annotations for *File* and *Directory* objects in the input object document. We recommend that annotations are appended on the same level as the standard fields (*class*, *location* and *format*), where the property is the key and the annotation itself the value. Values can be single annotations, arrays or (arrays of) dictionaries.

In addition, the authors can convey information about a *set* of input parameters via annotations in the root of the input object document (**IR6**).

Vocabulary
Information about a set of input values can be expressed under the *s:description* key.

For *File* and *Directory* inputs, we reused the Bioschemas Dataset profile v1.0[5], in this way complying with ?. ? shows how the Schema.org terms relate to the required metadata specified in **??** and **??**.

We recommend that authors adhere to this vocabulary when describing properties of their datasets which are domain-neutral. However, if they want to convey domain-specific information which is not covered by the terms in ?, they may choose to extend this annotation scheme with domain-specific ontologies (such as EDAM [20]), in this way fulfilling **IR5**.

Below, we show some annotation examples. In general, we recommend that authors provide at a minimum the metadata which is not covered by the identifier of the data they use (**IP4**).

5 https://bioschemas.org/profiles/Dataset/1.0-RELEASE

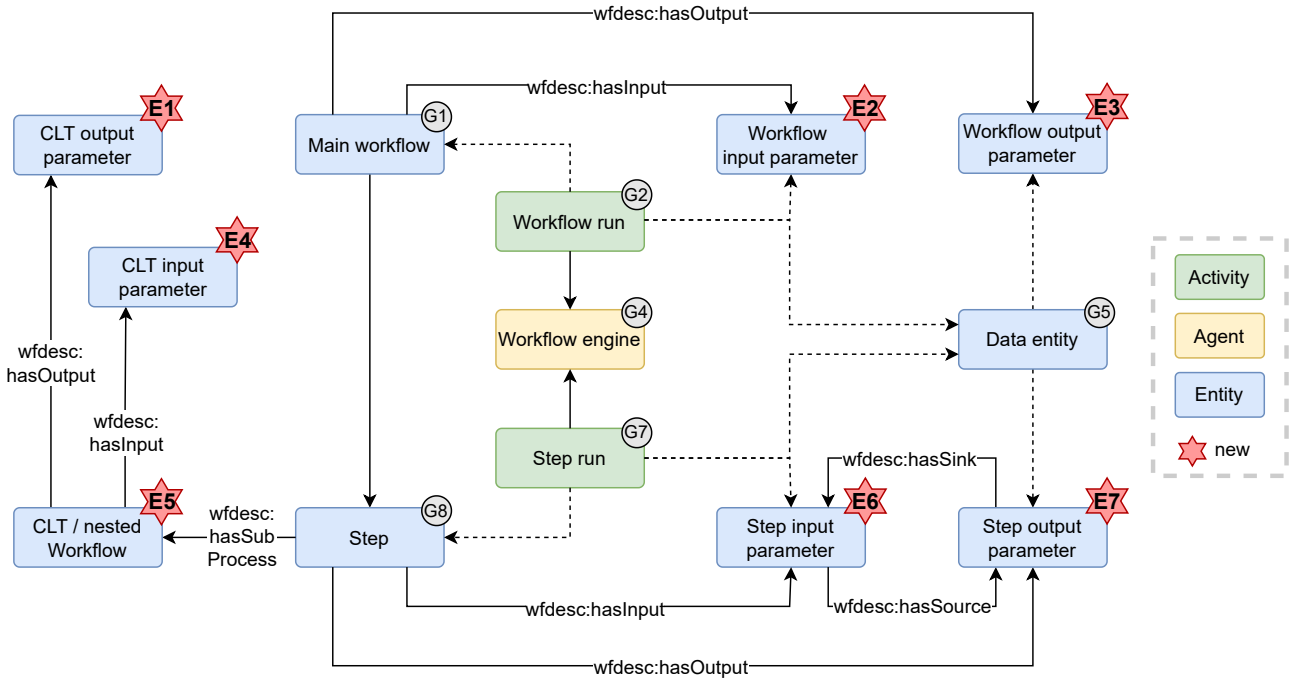

Figure 4. The RDF provenance graph, now extended with *CommandLineTools* and parameter entities. Red stars mark nodes which are part of the design extension. Parameters are linked to their *Workflow*, *CommandLineTool* or step via *wfdesc:hasInput* and *wfdesc:hasOutput*. Node G5 represents both input and output data entities. The data flow between steps is represented via *wfdesc:hasSink* and *wfdesc:hasSource*. Steps are linked to their underlying tools via *wfdesc:hasSubProcess*.

Design of an extension of the CWLProv provenance graph

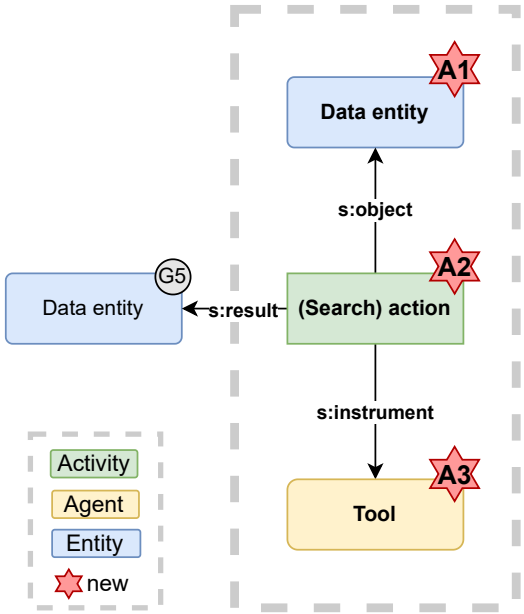

Figure 5. Actions represented in RDF provenance graph. Workflow input datasets (**G5**) are connected to the Actions (**A2**) which produced them via *s:result*. In addition, Actions can have properties such as *object* (**A1**) and *instrument* (**A3**).

In ?, we analyzed CWLProv for the representation of the provenance taxonomy we defined in ?. Based on this analysis, we concluded that not all provenance components were sufficiently represented in a structured format. Specifically, we found that although certain metadata was part of *primary-job.json* and *packed.cwl*, these annotations were not represented in RDF format.

In this section, we propose an extension of the design of the CWLProv provenance graph described in ?, which can at least represent the values for already supported metadata fields and can also be extended later with other metadata. In this way, we (partially) answer ??.

Here, we are only concerned with the representation of this information, if it exists. Mechanisms for the collection of the metadata (whether it be manually or automated), are considered out of scope for this thesis.

In , we first describe the requirements and principles which we used in this design. Subsequently, in , we give a high-level overview of the design. We then give recommendations for specific terms and vocabulary which can be used for the metadata fields *doc*, *label*, *intent*, and *format* which are part of v1.2 of the CWL Standards (). We describe how we have partially realized this design in *cwltool* (). Finally, we perform a conceptual analysis in and discuss the design in ??.

Requirements and principles

For this design, we reuse principles **IP1** and **IP2**. In addition, the RDF extension should adhere to four requirements.

PR1 Represent the annotations that are supported in CWL workflows according to the CWL standards v1.2, summarized in 5.
PR2 Represent the annotations for *Files* and *Directories* proposed in .
PR3 Represent the annotations for a collection of input values proposed in .
PR4 Represent the annotations for Actions proposed in ??.

High-level overview

In this section, we present a high-level overview of the extended provenance graph and link it to the requirements defined in .

Representation of CWL metadata fields.
4 shows how we extended the provenance graph to represent all current metadata fields (**PR1**). **In the new graph, every workflow component in 5 is represented as a distinct entity, and interrelated with terms from the *wfdesc* ontology (IP1).** We added entities describing the CWL tools that are run by the steps (**E5**) and linked them to their input (**E4**) and output parameters (**E1**) via *wfdesc:hasInput* and *wfdesc:hasOutput*. We connected the steps (**G8**) and tools via *wfdesc:hasSubProcess*. In addition, we linked the step parameters (**E6**, **E7**) together via *wfdesc:hasSource* and *wfdesc:hasSink* to express the data flow between the steps.

In CWLProv 0.6.0, the execution of nested workflows is described in separate RDF documents. **Following the same strategy, we recommend to only represent top-level parameters of nested workflows in the primary provenance graph.** More detailed annotations can be represented in the separate RDF documents which describe the nested workflows.

Representation of input data annotations.
This structure depicted in 4 also supports representation of the input data annotations described in (**PR2**), by transferring them to the data entities they describe (**G5**).

Representation of configuration settings annotations.
Annotations describing a collection of parameters (**PR3**) are specific to the workflow execution. Therefore, we recommend that these annotations are transferred to the entity in the provenance graph describing the workflow execution (**G2**).

Representation of input data history.
In the annotation scheme described in **??**, the history of (processed) input data is expressed through *Actions*. 5 presents their representation in RDF (**PR4**). Actions (**A2**) are performed on objects, which are data entities not aggregated in the CWLProv RO (**A1**). Actions can have additional properties such as Tools (**A3**) or other annotations as summarized in 4. They are connected to input data entities (**G5**) via *s:result*.

Details of the design

Here, we move from the structure of the provenance graph to the annotations that now can be attached to the newly added entities and with which terms they should be represented.

First, we describe which terms to use for the CWL-specific metadata fields *doc*, *label*, *format*, and *intent*.

Although we could use the exact terms with *cwlprov* prefix, we aim for interoperability with other provenance representations, such as the RO-Crate specification. Therefore, we reuse Schema.org terms, and because this is in agreement with our annotation scheme:

- **doc**: http://schema.org/description
- **label**: http://schema.org/name

  Renske: @Michael, should we change this now Simone doesn't find this a good term for RO-Crate?
- **format**: http://schema.org/encodingFormat
- **intent**: http://schema.org/featureList

In addition, we recommend that custom annotations attached to workflow components and input objects are transferred as they are. We make an exception for *s:additionalType*, for which the value can be added to the *types* of the entity it describes, instead of being literally transferred as *s:additionalType*.

When entities are described with nested annotations (e.g. *s:citation*), we recommend to make this a separate entity in the graph instead of a nested annotation, in compliance with the PROV data model (**IP1**).

Partial realization in *cwltool*

Renske: Update, when this is published there should be full support in cwltool.

Because of time constraints, we could only partially realize the provenance graph extension in *cwltool*. At the time of writing, annotations directly associated with inputs of type *File* and *Directory*, as exemplified in **??**, are propagated to the RDF provenance record. However, annotations of the collection of input parameter values (**??**) or annotations under *cwlprov:prov* (**??**), are currently not represented in RDF.

Conceptual analysis of the design extension

Renske: leave in?

To test the extended design, we analyzed the RDF provenance graph which would have been associated with the epitope prediction workflow we used as an example in this thesis. The elements of the design which were not yet realized in *cwltool*, we emulated via manual annotations of the document.

SPARQL queries.

Renske: replace with all sparql queries

Here, we present two example SPARQL queries we issued on the emulated extended provenance graph. The first (**Q1**) extracts the DOIs of all publications which were the citations of the used inputs.

The second (**Q2**) lists the formats for every file for which this is specified.

SPARQL queries

List all Docker images used in this workflow run.

```
PREFIX cwlprov: <https://w3id.org/cwl/prov#>
PREFIX wf: <arcp://uuid,a914217a-5cd2-457d-85cc-7472eeb17bfd/workflow/packed.cwl#>
PREFIX wfdesc: <http://purl.org/wf4ever/wfdesc#>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT ?step ?image

WHERE {
    wf:main a wfdesc:Workflow ;
            wfdesc:hasSubProcess ?step .
    ?step_execution prov:qualifiedAssociation ?association .
    ?association prov:hadPlan ?step .
    ?step_execution prov:wasAssociatedWith ?agent .
    ?agent cwlprov:image ?image .
}
```

List every entity that has a DOI.

```
PREFIX s: <http://schema.org/>

SELECT ?doi
WHERE {
    ?aname s:identifier ?doi .
    ?entity s:citation ?aname .
}
```

Extract the formats of all files for which this is specified.

```
PREFIX s: <http://schema.org/>
PREFIX cwlprov: <https://w3id.org/cwl/prov#>
PREFIX prov: <http://www.w3.org/ns/prov#>

SELECT DISTINCT ?data_entity ?basename ?format

WHERE {
    ?data_entity s:encodingFormat ?format .
    ?id prov:specializationOf ?data_entity .
    ?id cwlprov:basename ?basename .
}
```

Extract description of workflow run.

```
PREFIX s: <http://schema.org/>
PREFIX wfprov: <http://purl.org/wf4ever/wfprov#>

SELECT ?desc

WHERE {
    ?id a wfprov:WorkflowRun .
    ?id s:description ?desc .
}
```

Extract metadata of workflow.

```
PREFIX s: <http://schema.org/>
PREFIX wfdesc: <http://purl.org/wf4ever/wfdesc#>

SELECT ?id ?doc ?intent

WHERE {
    ?id a wfdesc:Workflow .
    ?id s:description ?doc .
    ?id s:featureList ?intent .
}
```

List all the steps of the workflow, with metadata.

```
PREFIX s: <http://schema.org/>
PREFIX wfdesc: <http://purl.org/wf4ever/wfdesc#>

SELECT ?step_id ?doc ?label ?clt

WHERE {
    ?wf a wfdesc:Workflow .
    ?wf wfdesc:hasSubProcess ?step_id .
    OPTIONAL { ?step_id s:description ?doc . } .
    OPTIONAL { ?step_id s:name ?label . } .
    OPTIONAL { ?step_id wfdesc:hasSubProcess ?clt . } .
}
```

List all the input parameters of 1 particular step.

Renske: Make this general for any workflow, not tailored to this particular RO.

```
PREFIX s: <http://schema.org/>
PREFIX cwlprov: <https://w3id.org/cwl/prov#>
PREFIX prov: <http://www.w3.org/ns/prov#>
PREFIX wfdesc: <http://purl.org/wf4ever/wfdesc#>

SELECT DISTINCT ?step_input

WHERE {
    ?wf wfdesc:hasSubProcess <arcp://uuid,eb41f41c-d7b4-4999-9ce9-719fdc8c12b1/workflow/packed.cwl#main/combine_labels> .
    OPTIONAL { <arcp://uuid,eb41f41c-d7b4-4999-9ce9-719fdc8c12b1/workflow/packed.cwl#main/combine_labels> wfdesc:hasInput ?step_input }
}
```