





Master Thesis

**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**

Author: Renske de Wit  (2679596)

Supervisor: Alexandru Iosup 
Daily supervisor: Michael R. Crusoe 
Examiner: Sanne Abeln 

*A thesis submitted in fulfillment of the requirements for
the Minor Research Project (XM_0072, 30 EC) of
the joint UvA-VU Master of Science degree in Bioinformatics*

August 23, 2022

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
Vrije Universiteit Amsterdam
h.c.de.wit@student.vu.nl

Michael R. Crusoe
Vrije Universiteit Amsterdam
michael.crusoe@vu.nl

Alexandru Iosup
Vrije Universiteit Amsterdam
a.iosup@vu.nl

ABSTRACT

In the era of big data and big science, workflows have been proposed as a means to achieve computational reproducibility. However, sharing the results of a workflow execution is challenging. Research Objects (ROs) can solve this problem by packaging the workflow description and the data associated with an analysis, together with structured metadata describing the meaning and relationship between the contained entities. ROs contain the provenance of the generated results. However, there is a need for a standard of metadata which should be contained in ROs. Here, we define a taxonomy of provenance types based on a use case Bioinformatics workflow. Subsequently, we assess CWLProv 0.6.0, a standard for sharing the execution results of CWL workflows as a RO, for the representation of each of the components of the taxonomy. Based on the results of this analysis, we propose a standard for the annotation of input data, as well as an extension of the provenance graph to enable richer annotations. We are confident that the work described here is not only relevant for CWLProv but can also be applied to other RO specifications (such as the Workflow Run RO-Crate profile). A standardized approach to the representation of workflow executions can serve as the basis for automated extraction of the collected provenance. Ultimately, this is one step closer to achieving computational reproducibility.

1 INTRODUCTION

In the era of big data and big science, *computational reproducibility* is an important means to verify the validity of scientific conclusions drawn from raw data. However, across multiple domains of science, computational reproducibility is often not achieved due to lack of transparency or code rot [8]. Workflow-centric research objects have been proposed as a method to improve the transparency of computational analyses [3]. In addition to the workflow description, ROs may encapsulate relevant resources such as data and software, together with structured annotations describing these entities and their relations to each other. However, which resources should be contained and how they should be described is still an open challenge.

Science influences many aspects of our lives. Decisions and policies are often motivated and defended with an appeal on science. Because science is so influential on the health and well-being of entire populations, the ability to assess the validity of scientific conclusions is extremely important.

Reproducibility is an important scientific principle. It is based on the conviction that human capacity to discover the rules of nature is limited [8]. Every scientific conclusion that is made has a degree of uncertainty associated with it, and belief in its validity is strengthened if repeated observations produce consistent results.

In this work, we consider a specialized type of reproducibility known as *computational reproducibility*. Nowadays, science is strongly data-driven. Experiments involving cutting-edge technologies generate large volumes of data, which can only be understood through a series of operations and transformations and subsequent statistical analyses. In contrast to the experiments generating the data, which always deal with unaccounted sources of variation, reproducibility of the computational analysis (i.e. obtaining consistent results given the same input data and computational methods [8]), should in principle be feasible. However, retracing the path from the raw data to the final results without knowing the intermediate steps (the *provenance* of the results) can be extremely complicated, if not impossible.

Although conceptually possible, a range of studies across different domains of science have concluded that *computational reproducibility is often not achieved in practice*. Some studies (e.g. [40][42]) observed a lack of availability of data and code, i.e. insufficient *transparency* of the research. When scientific findings are not reported transparently, others do not have the means to reproduce the analysis. Other studies (e.g. [7]) encountered a phenomenon colloquially known as *code rot* [8]. In this case, the original code can not be executed anymore, or generates different output than the original analyses due to updates to the underlying software and its dependencies.

Workflow thinking [19] can help to improve computational reproducibility. In this method, a (computational) process is divided into a series of steps, with the output of one step becoming the input of another. Workflows can be visualized as a directed graph, where the nodes are the operations and the edges the data flows between them.

Figure 1 envisions a future in which workflows are an integral part of reproducible computational research, central to an ecosystem of tools and standards which have emerged over the last decade. In this scenario, workflows are executed by *workflow systems*, which leverage efficient job-scheduling and containerization technologies, while collecting relevant metadata about the provenance of the computed results. In addition, *workflow registries* provide a medium to share and find workflow descriptions easily. Another important player are FAIR [46] *data repositories*, both for publishing the results of a workflow execution and finding the data to use as input. After

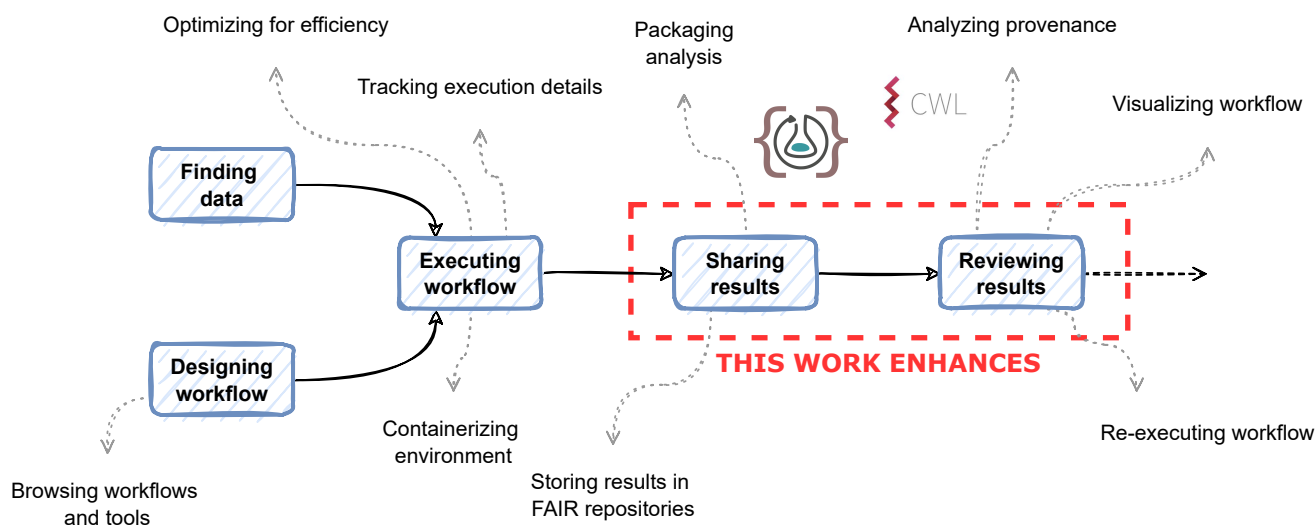


Figure 1: A vision of an ecosystem of workflow-related resources and its benefits for the reproducibility of computational processes. In this vision, workflows and data are stored in FAIR repositories, are executed in containerized environments by workflow systems which track details of the computation, and enable sharing of the analysis in a structured format (such as ROs following the CWLProv or RO-Crate specification). The packaged analysis can later be used to understand the details of the analysis or re-execute the workflow. Fundamental to the success of this approach is the interoperability of the components of the ecosystem.

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.

In this work, we are concerned with the format in which the results of the analyses are shared. To make the vision presented in Figure 1 a reality, the components of the ecosystem need to be compatible with each other. In practice, there are a number of challenges to overcome.

Firstly, the wide variety of workflow languages (which are sometimes workflow system-specific) compromises the portability and interoperability of workflows between different systems. To address this issue, Common Workflow Language [9] aims to provide a standard for workflow descriptions which can be executed by multiple workflow systems.

Secondly, there is currently no widely accepted standard for representing the results of a workflow execution. This representation should both be used to infer the provenance of the data generated by the workflow, and be machine-accessible (because the results should be easily findable and storable).

Research Objects (ROs) [3] have been proposed as a packaging solution for packaging the workflow description, with the input configurations as well as details about the execution itself. ROs are data structures which aggregate all the data entities and parameter values involved in a computational analysis, with the workflow description and metadata files describing the relationships between the data entities contained in the RO in a machine-readable format.

In this project, we investigate CWLProv, a specification for the packaging of CWL workflow executions as an RO. Related specifications also exist, such as the RO-Crate Workflow Run profile [38].

1.1 Problem statement

However promising ROs are as a solution to the computational reproducibility crisis, there are still several open problems which need to be addressed.

Open challenges in the field include the large size of ROs due to their inclusion of all data involved in the workflow execution, or how to automate the collection of relevant metadata by workflow systems.

The goal of this thesis is to address two other challenges: *Which metadata to include to make workflow executions sufficiently transparent*, and *How to represent this information in a structured format*.

Finding the answers to these questions is conceptually challenging, because the details which constitute provenance can vary widely between scientific domains. It is difficult to define a standard of metadata which is on the one hand broadly applicable across many disciplines, while still able to express domain-specific information on the other hand.

1.2 Research questions

In this thesis, we address the following research questions:

RQ1 Which elements of a Bioinformatics workflow execution should be described in RO metadata, considering representative use cases of their provenance? The use

cases of ROs and their provenance influence which metadata is important to capture and store. There may be use cases which apply to most scientific workflows, while others are highly domain-specific. Separating between these can be non-trivial.

RQ2 How does provenance contained in ROs following the CWLProv 0.6.0 specification compare to the elements defined in answer to RQ1? It is important to assess how close CWLProv ROs are to the ‘ideal’ standard of metadata which is necessary in practice. Such an analysis is non-trivial, because it should discriminate between different forms of representation (structured or unstructured), and take into account how much effort is required to supply the information.

RQ3 How to improve the status of CWLProv provenance such that it meets the requirements defined in the answer to RQ1? After having identified elements where CWLProv 0.6.0 does not adhere to the requirements, it is important to consider how this information could be represented in future versions of CWLProv. Careful selection of models and ontologies should be performed, considering how they fit in with the current CWL Standards, and still be extendable with other metadata which may be added in the future.

1.3 Approach

To answer the research questions, we take the following (inter-disciplinary) approach, integrating methods from Bioinformatics, Computer Systems and Software Engineering.

We answer RQ1 by defining a taxonomy of provenance based on representative questions about the provenance of an example Bioinformatics workflow. Firstly, we implement a workflow for epitope prediction in CWL, considering potential challenges for its reproducibility. Secondly, we identify a set of use cases for which the provenance of the workflow may be queried in the future, and formulate questions associated with each use case. Thirdly, we categorize the questions into 6 elements of the workflow execution which should be represented in its provenance. We relate them to existing standards and recommendations for data [12], [39] and software citation [35], which are both based on the FAIR principles [46].

Although other models of provenance exist (e.g. the PRIMAD model, [19]), our taxonomy is grounded in real life use cases of a specific Bioinformatics, and specifies in detail which metadata is necessary to address these use cases. Other provenance taxonomies, such as [32], are concerned with features of provenance systems, not with the content of the provenance they collect.

To answer RQ2, we analyze the representation of each component of the taxonomy in CWLProv 0.6.0. We distinguish between representation in RDF, structured but CWL-specific documents, and unstructured representation.

Based on the results of the analysis, we answer RQ3 in two ways. For both, we adhere to the AtLarge design framework [22]. Firstly, we design and implement a model for the structured annotation of workflow input data. This scheme uses terms from

the Schema.org¹ [20] ontology and aligns with the *Dataset* profile established by the Bioschemas community [30]. The proposed model can be used to express information which is relevant in most scientific domains, and can also be integrated with terms from other ontologies for the representation of domain-specific metadata.

Secondly, we design an extension to the RDF provenance graph for support of richer annotations. Taking the existing design into account, we extend the model with terms from the *wfdesc* ontology [36], and propose terms by which CWL-specific metadata fields and other structured annotations (following our annotation scheme) can be represented in the provenance graph.

1.4 Main contributions

The work described in this thesis includes several contributions, both conceptual and technical.

- C1 Technical:** Implementation of a workflow for epitope prediction in CWL (Section 3).
- C2 Conceptual:** A taxonomy of provenance types, together with the metadata required to describe them (Section 4).
- C3 Conceptual:** A qualitative analysis of CWLProv provenance, evaluating the representation of the elements of the provenance taxonomy (Section 5).
- C4 Conceptual:** The design of an annotation scheme for input data of CWL workflows (Section 6).
- C5 Conceptual, Technical:** An extension of the design of the CWLProv provenance graph and its partial realization in the CWL reference engine *cwltool* (Section 7).

1.5 Reading guide

Figure 2 shows an overview of the structure of this thesis. In this section, we introduced ROs as a potential solution to the problem of computational reproducibility. Section 2 gives a brief introduction into CWL workflows and the description of their execution in CWLProv ROs. The next section (Section 3) explains the rationale for epitope prediction, gives a conceptual overview of our example workflow, identifies key challenges for its computational reproducibility and explains how we address these challenges in its CWL implementation. In Section 4, we outline the taxonomy of provenance types and connect them with metadata which describes them. In Section 5, we evaluate the representation of each of these provenance types in CWLProv. Section 6 outlines an annotation format for input data which authors of CWL workflows can use to enrich their workflow descriptions. In Section 7, we describe how CWLProv can be extended to solve some of the gaps we found in the analysis. Finally, we conclude the thesis in Section 8.

2 BACKGROUND: CWL AND CWLPROV

In the previous section, we introduced the problem we address in this thesis, and provided a high-level overview of our approach. In this section, we define the concepts we introduced in the Introduction in more detail, and provide further background information which is necessary to understand the rest of the thesis. We start

¹<http://schema.org/>

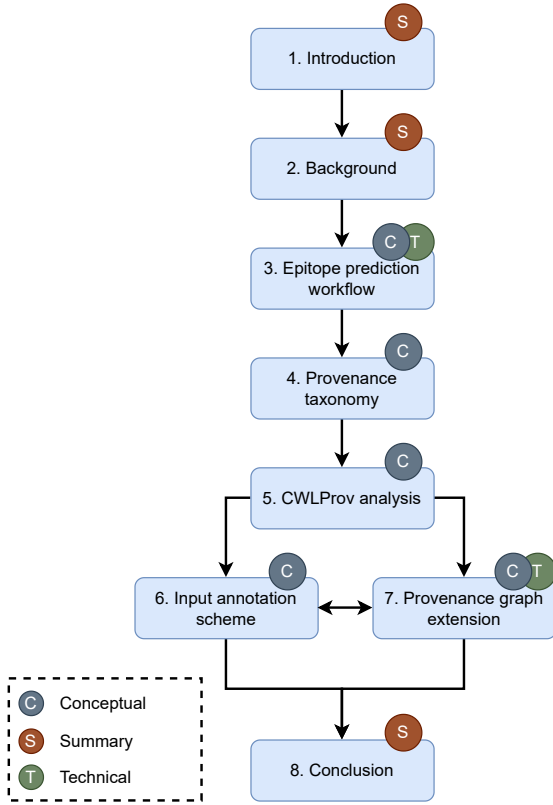


Figure 2: The structure of this thesis. The arrows denote suggested reading flows. The design of this figure was inspired by [1].

with a short introduction of CWL workflow descriptions in Section 2.1. Subsequently, in Section 2.2, we explain how the results of CWL workflow runs are packaged in CWLProv ROs.

2.1 The Common Workflow Language Standards v1.2

In this section, we give a primer into the background of CWL, sufficient to understand the rest of this thesis. A full explanation of the CWL standards is considered out of scope for this work and can be found in [9].

2.1.1 Motivation for the CWL standards. CWL was developed for two main reasons:

- **To improve the portability of workflows.** *Portability* means that the workflow can be taken from the platform (e.g. the operating system or workflow engine) on which it was originally developed and executed on another platform, producing the same output. CWL aims to be a unifying language which can be executed on multiple workflow systems.
- **To provide a standard for good workflow descriptions.** Since the concept of workflow thinking can be implemented

in many different ways, there need to be formalized guidelines for writing high-quality workflows. Shell scripts and workflows on highly sophisticated workflow systems may produce the same output, but the quality of one can be much higher, because it also describes required resources, is easily rerunnable with different inputs (because it avoids the use of hard-coded filepaths), contains information about the underlying software which are used in the steps, etc. The CWL standards formally specify essential components of workflows and how they should be described.

CWL workflows are run with a workflow description file and an input parameter file.

2.1.2 CWL Workflow description. CWL workflows consist of a series of steps, which each run a *CommandLineTool* or nested *Workflow*. *CommandLineTools* are CWL descriptions of command-line programs. The syntax of *Workflows*, *CommandLineTools*, and other CWL documents is built upon YAML.

Parameters: Steps have input parameters, which can be either connected to the output parameters of previous steps, or to workflow-level *WorkflowInputParameters*. In addition to input parameters, workflows can also have one or multiple *WorkflowOutputParameters*, which are linked to the *WorkflowStepOutputs* which produce them.

The CWL standards distinguish between different types of parameters, such as *File*, *Directory*, *string*, or *float*. Parameters can also be arrays. The expected parameter types should be explicitly defined in the *CommandLineTool* and *Workflow* descriptions, and must correspond between steps that are linked together.

Requirements: *Workflows* and *CommandLineTools* can also specify resource requirements, both hardware (memory, storage, or CPU cores) and software. In addition, *DockerRequirement* indicates that a step should be run inside a software container, and specifies the repository from which the image can be pulled or the Dockerfile from which it can be built.

Metadata: Finally, both *Workflows* and *CommandLineTools* can be enriched with annotations to make them and their components easier to understand. There are CWL-specific metadata fields (*doc*, *label*, and *intent*), but custom semantic annotations are also supported. Table 1 summarizes which components can be annotated and their associated metadata fields.

2.1.3 CWL input object document. The input object is a data structure which contains the values assigned to the workflow input parameters. Although parameter values can be specified at the command line, they are usually presented in a file.

The input object is an array, where the keys are the parameter names and the values are the parameter values.

The values for all parameter types except *File* and *Directory* are simply the values. For files and directories, however, the value is an object with multiple fields, including the location of the file or directory. In addition, the format can be specified, as well as custom semantic annotations.

2.1.4 Execution of CWL workflows. It is important to realize that CWL is a language, not a workflow system itself. Currently, support for the CWL standards has been implemented in at least 5 workflow runners and platforms [9]. These workflow systems may

have implemented some or all features of CWL. They also differ in the way they execute workflows: Toil is designed to execute computationally heavy workflows efficiently. Some execute their workflows in the cloud. In this project, we use the CWL reference implementation *cwltool*.

Table 1: 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	•	•		•

2.2 CWLProv 0.6.0

In the previous section, we explained that CWL is a standard for workflow descriptions. In this section, we explain CWLProv, which is a standard for the description of (CWL) workflow *executions*.

2.2.1 Motivation for CWLProv. The *provenance* (i.e., lineage) of the results of a computational analysis can not be represented by the workflow description alone. It is also dependent on the input data (parameter values) and the computational environment in which the analysis was performed. Although it is conceptually possible to share this information separately, the complexity of the analyses (some workflows are executed on distributed systems) can render this practically infeasible. In addition, workflows may be executed multiple times with different settings, and can themselves be subject to change, and tracing the generated results back to the exact version and settings of the workflow that produced them can be non-trivial.

The CWLProv specification addresses a need for the representation of workflow executions which connects the data, workflow and computational environment in a machine-accessible format.

2.2.2 The CWLProv 0.6.0 specification. CWLProv ROs contain the CWL workflow description, the input object and all data products (inputs, intermediate and final outputs) associated with a given workflow execution. In addition, they also contain RDF files detailing the relation between the aggregated resources and the workflow execution itself.

In principle, sharing just an RO is sufficient to reproduce the analysis on another system and generate the same output [28]. A tool which can be used for this is *cwlprov-py* [28], which takes a CWLProv RO as input and can rerun single steps or entire workflows.

The following sections explain the structure and purpose of specific metadata files in CWLProv ROs in more detail.

2.2.3 Packed.cwl. The entire workflow description packaged into one document (as opposed to separate *Workflow* and *CommandLineTool* files). The document contains all the structured information from the original workflow description, including metadata, but unstructured information (e.g. comments) is lost.

2.2.4 Primary-job.json. The input object document. It contains the values for all parameters in the original input file, but not for parameters with default values (specified at *CommandLineTool* or *Workflow* level and preserved in *packed.cwl*).

2.2.5 Manifest.json. An RDF document in JSON-LD format, listing all data entities aggregated in the RO.

2.2.6 Provenance graph. One or more documents in RDF format (stored in the *provenance* directory) describing the relationships of the aggregated data entities with the workflow execution.

To express provenance, CWLProv uses PROV-DM [31], a standard for the RDF representation of provenance which is applicable to most domains of science. In PROV, all provenance relationships are (in general) expressed with a combination of just three types: *Activities*, *Entities*, and *Agents*. *Entities* are derived from other *Entities* via *Activities*, which are controlled by *Agents*.

Figure 3 shows a high-level overview of the CWLProv provenance graph. Summarizing, the graph describes the execution (G2) of a workflow description (G1) by a workflow engine (G4), which was initiated by a human agent (G3). The workflow description consists of one or multiple steps (G8) and their execution (G7) can occur in a software container (G6). During the execution, data entities (G5) are consumed or produced and are linked to the step and workflow input and output parameters.

Although all components in the graph and their relations can in principle be expressed as a PROV type or property, its vocabulary is often not specific enough to represent subtle differences between different types of interactions and relationships (workflows and files are both *Entities*, yet their distinction can be quite important). Therefore, CWLProv also uses other ontologies with workflow-specific vocabulary, such as *wfdesc* [36] and *wfprov* [37]. *Wfdesc* expresses concepts related to workflow *descriptions*, whereas *wfprov* conveys details about their *execution*. In addition, CWLProv uses terms from several other ontologies, including Schema.org² [20].

²<http://schema.org/>

An alternative to determining the protein structures experimentally is predicting them computationally. Given that protein structure ultimately arises from sequence (see *A primer on protein structure*), epitope prediction models like SeRenDIP-CE [21] attempt to identify the epitope residues based on features inferred from protein sequence. The predictions made by these models can then be used to preselect antibodies which are likely to bind to the epitopes of a given antigen, enabling a much more targeted approach than evaluating each candidate experimentally.

However, a challenge which epitope prediction models face is the scarcity of training data, which compromises the accuracy of the predictions for proteins which do not have an experimentally resolved protein structure.



A primer on protein structure

Proteins consist of one or multiple chains (peptides) of linked-together amino acids. The order and identity of these amino acids (the *protein sequence*) is specified in the DNA. Each amino acid has distinct characteristics. Some prefer to interact with water on the protein surface (hydrophilic), others are hydrophobic and are often located on the inside of the protein.

We discriminate between four layers of protein structure. The *primary structure* is the protein sequence. The *secondary structure* arises from hydrogen bonds within the peptide backbone. On top of that, the peptide can be folded into the *tertiary structure*. The *quaternary structure* arises when the folded protein chain interacts with other (folded) protein chains to form a complex.

Summarizing, we see that even though protein structures can be very complex, they ultimately arise from the information encoded in the DNA.

3.2 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, explained in Section 3.3.
- WR2** Address challenges which compromise the transparency and reproducibility of this workflow, explained in Section 3.4.

3.3 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 a use case 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 [6]. In its turn, that model was derived from OPUS-TASS [47], 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.**

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

- **Protein Data Bank (PDB)** [5], a database of experimentally resolved structures of proteins and complexes.
- **Structural Antibody Database (SABDab)** [13], 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** [43], a dataset of protein sequences containing annotations for general PPI interactions, previously derived from structure.
- **HHBlits reference database.** Used by HHBlits [33] to compute some of the input features.

3.3.2 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 [25] 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.

3.3.3 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 physico-chemical 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.

3.4 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.

3.4.1 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).

3.4.2 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.

3.4.3 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 [13].

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 tool³ [44] may not return the same mappings in the future.

3.5 Implementation of the workflow

In this section, we describe how we implemented the epitope prediction workflow in CWL, considering the requirements described in Section 3.3 and Section 3.4.

Figure 4 shows an overview of the CWL implementation of the workflow, which was based on a diagram provided by the workflow authors (Figure 4a). Figure 4b shows an early version of the workflow, and Figure 4c the final implementation.

3.5.1 Implementation of the conceptual method (WR1). To address the first requirement, the workflow starts by issuing a query to the PDB Search API⁴ (*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*).

3.5.2 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.

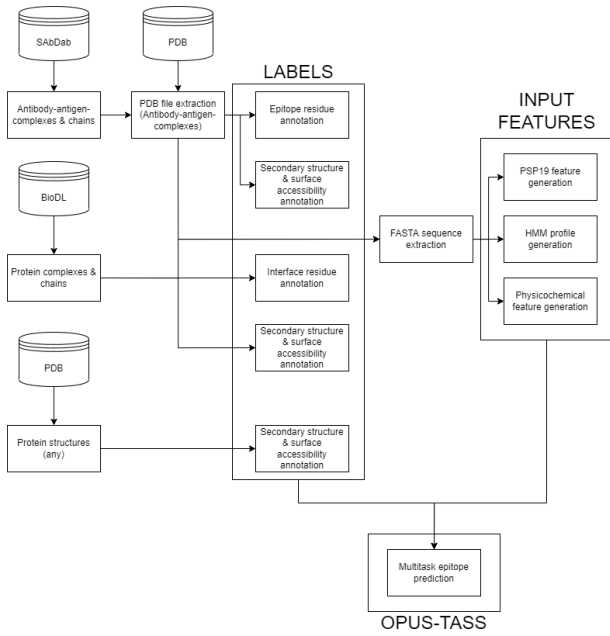
To increase the portability of the computational environment, some steps are executed inside software containers. The workflow engine pulls Docker images from external repositories and converts them to Singularity [29] containers, the software which is installed on the Bazis HPC cluster on which we executed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

3.5.3 Emulation of workflow steps. 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.

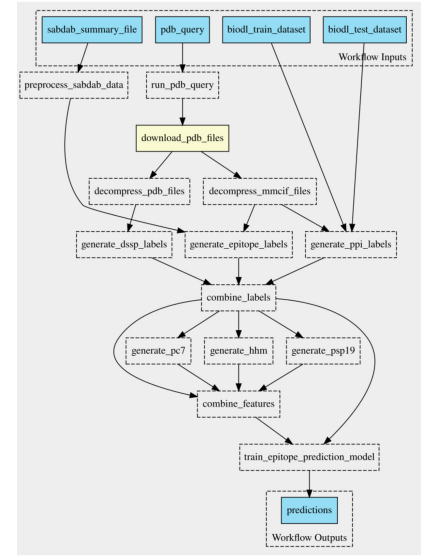
In summary, for three steps we used scripts provided by the workflow authors (*preprocess_sabdab_data*, *generate_epitope_labels*, and *generate_dssp_labels*). For other steps we reused or modified code from OPUS-TASS (*generate_pc7*, *generate_psp19*, and *generate_hhm*) or other sources (*run_pdb_query*, *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/>

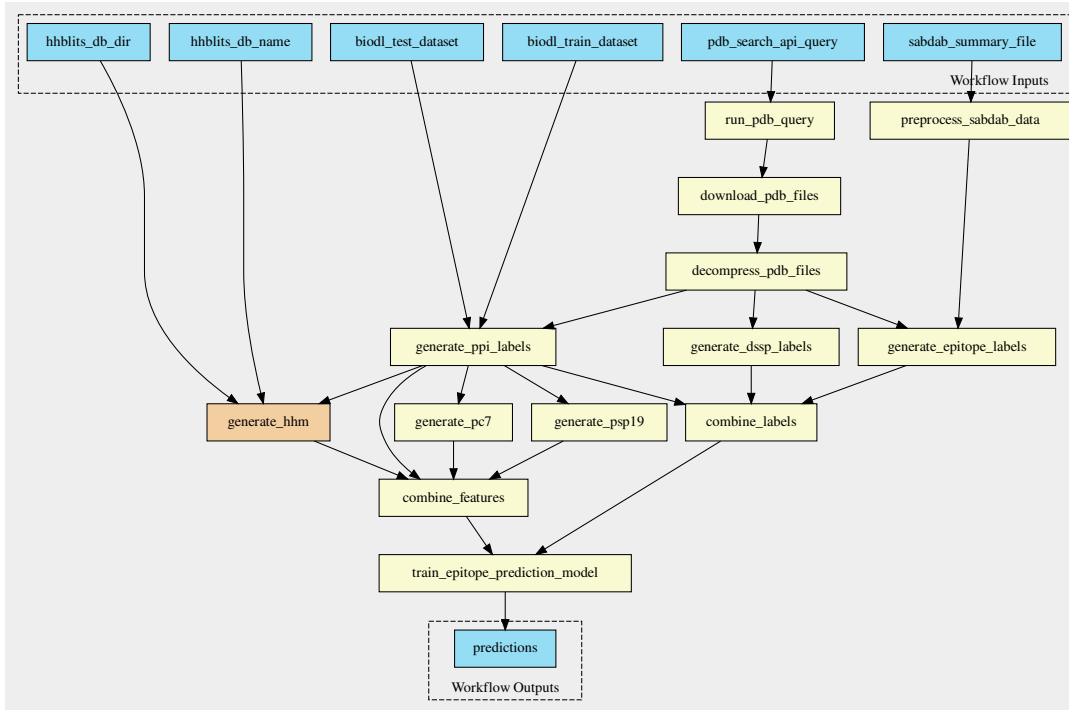
⁴<https://search.rcsb.org/index.html#search-api>



(a) Diagram of the conceptual method, provided by the workflow authors.



(b) Early version of the implementation, in which most steps are still abstract *Operations* (dashed nodes).



(c) The final implementation of the workflow.

Figure 4: Progression of the workflow implementation from initial sketch to full implementation. 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.

3.6 Discussion

In this final section, we discuss the implementation of our example workflow. A number of steps still need to be taken before this model can be used to generate biologically meaningful epitope predictions.

Firstly, we focused on the high-level design of the workflow, and organized this in such a way as to maximize the transparency and reproducibility of the workflow in the future. We only addressed challenges for reproducibility which we anticipated given the nature of the method, hence there might be more challenges when the workflow is reproduced at some point in the future.

Secondly, the design of the workflow only considers the training of the model. In the future, extra steps could be added which split the dataset into training and test set, and potentially a separate workflow can be made in which the trained model computes predictions for query sequences. In addition, future work should replace the emulated steps with functional code and test them with (OPUS-TASS) reference data.

Finally, not all steps were containerized, because we could not find suitable images for all steps, and Singularity is incompatible with *dockerFile* (a local Docker recipe). In the future, these steps can be containerized if images are built from Dockerfiles on a system which has Docker installed, and uploaded to our own repository. Because of time constraints, we could not do this ourselves, but we included the Dockerfiles as comments in the *CommandLineTool* descriptions.

4 DEFINITION OF A TAXONOMY OF PROVENANCE

In Section 3, we described the implementation of a workflow which we use as an example. In this section, we synthesize requirements for CWLProv provenance, in this way answering **RQ1**. Section 4.1 specifies use cases for the provenance which are specific for this workflow. In Section 4.2 we link this to 6 aspects of the workflow and its execution, which should be represented in provenance. In Section 4.3, 4.4, 4.5, 4.6, 4.7, and 4.8, we elaborate on each of these aspects and explain by which metadata they should be accompanied to fulfill the requirements associated with the use cases described in Section 4.1.

In contrast to other provenance models, our taxonomy is both based on real-life use cases of the provenance of a specific Bioinformatics workflow, and connects them with existing standards for data and software citation to identify which metadata are necessary to fulfill the use cases.

4.1 Workflow use cases

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 5 use cases, each in a different stage of the workflow's lifetime.

- 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 (allowing 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**. The 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. The RO is the standard output of the web-based tool.

4.2 Synthesis of a taxonomy of provenance

Based on the use cases (Section 4.1) we synthesized a list of questions, presented in Appendix A. Based on these questions, we propose a 6-component taxonomy of provenance types which should be represented in the provenance of the workflow execution.

- 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.
- T3 Software.** The tools directly orchestrated by the workflow, and their dependencies.
- T4 Workflow.** The workflow and tool descriptions, but not the software they control.
- T5 Computational environment.** Metadata about the system on which the workflow was executed, comprising both software and hardware.
- T6 Execution details.** Additional information about the workflow execution itself.

The rest of this section elaborates on these components. For every element of the taxonomy, we define its meaning, why it should be included, and how it should be represented.

4.3 T1: Definition of metadata for scientific context

In Section 4.2, we presented scientific context (**T1**) as one of the aspects that should be present in the collected provenance. In this section, we elaborate on what we mean with scientific reasoning and how it should be incorporated in provenance.

Why: Representing the scientific thinking process in provenance is meant to make the connection between the article advertising the research and the research itself, contained in the RO. This connection is bidirectional: ROs can be used as a guide during manuscript writing as well as for providing context for a third party who has read the paper and wants to understand the analysis in more depth.

What: The scientific context covers many aspects of the research, from the reasons why particular input data and parameter settings were chosen [8], to the design of the workflow (why particular steps were included), to the rationale for the study and the overall hypothesis of the research [4][19]. Even negative results can be reported and explored strategies which did not work can be included [41]. In addition, it can include the interpretation of the results and their implications on the field (or maybe the output of an intermediate step warrants the inclusion of another preprocessing step).

How: We classify scientific context into 3 components:

- 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.
- 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.

4.4 T2: Definition of metadata for data

In this section, we describe which metadata should be attached to data entities (**T2**) in the provenance record.

What: We mean not only input data, but also (intermediate) output data.

Why: Metadata associated with data entities can have at least two purposes:

- **To explain the meaning and context of the data.** The context of the data should be described, because others need to be able to assess the appropriateness of the data for the purpose of the computational analysis. *In the case of the epitope prediction workflow, it is important to understand the composition of the training set, since this is highly influential on the performance and applicability of the model.*
- **To describe data which is not contained in the RO.** To reduce RO size, or because the data cannot be shared for privacy reasons (it is proprietary or in other ways non-public), data may not be present in the RO but stored in an external repository. Characteristics of the data may still be included which provide information, to make sure that the data in the repository is the same as which was used

in the original analysis. *Use case U5 is an example where limiting the size of the RO may be desirable. Instead, the RO could contain references to datasets which are too large and are instead stored in an external repository.*

How: We identify 4 categories of data metadata which should be represented based on these two reasons. Here we link them to established recommendations and best practices for data citation, according to the FORCE11 Data Citation Principles [12], which are based on the FAIR principles [46].

- D1 Identification:** 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.
- D2 File characteristics:** Filename, format, creation and last modification timestamps, size, and checksum.
- D3 Access:** URL to a downloadable form of the data. License.
- D4 Mapping:** The workflow and step parameters for which the data is an input or output.

4.5 T3: Definition of metadata for software

In this section, we elaborate on the representation of software in the provenance record (**T3**).

What: In particular, we mean the command-line programs directly orchestrated by the workflow. However, these principles also apply to the workflow itself (Section 4.6), the workflow engine (Section 4.8), and computational environment (Section 4.7).

Why: There are two main reasons why software should be described with metadata:

- **Obtaining identical results in a re-execution may be highly dependent on the version of the tools used.** This is important when reproducing the workflow.
- **The original software may not be available or executable in the future.** In this case, the software should be sufficiently described for others to choose a suitable equivalent.

How: We identify 3 categories of software characteristics which should be included in the provenance. Here we link them to established recommendations and best practices for software citation, according to the FORCE11 Software Citation Principles [35], which were adapted from the FORCE11 Data Citation Principles.

- SW1 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.
- SW2 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 Access:** URL to downloadable, executable form of the software. License.

4.6 T4: Definition of metadata for workflow

In this section, we describe the metadata that is associated with the workflow (T4).

What: We define workflow here as the documents described in CWL (for other workflows, this would be the top-level script, not the underlying software that it controls). Hence the workflow comprises the main workflow description and the *CommandLineTool* and nested *Workflow* descriptions.

Why: Workflows provide both a high-level overview of the analysis as well as details which are difficult to convey in a textual format in the Methods section of a scientific article [16]. In addition, workflows are software which should be preserved and reused [17].

How: The metadata associated with workflows comprise general software metadata, in addition to workflow-specific information.

WF1 General software metadata: At workflow and step level, according to Section 4.5.

WF2 Parameters: Type, format, and description, at workflow and step-level.

WF3 Requirements: Software and hardware resources which are required to execute the workflow or workflow steps.

4.7 T5: Definition of metadata for computational environment

In this section, we describe the requirements for computational environment (T5).

What: Environment encompasses both software and hardware infrastructure, and may be part of a software container.

Why: Information about the computational environment is important for reproducing the workflow (U4).

- **Required resources:** Details about the resources available on the system which executed the original analysis can give an indication of what is necessary to rerun the workflow, even if this is not documented in the workflow description.
- **Debugging:** The generated output (or executability) of a step may be sensitive to specific versions of the tool orchestrated by the step or its dependencies.

How: We discriminate between three components of the computational environment:

ENV1 Software: 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 Section 4.5.

ENV2 Hardware: 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 Section 4.5.

4.8 T6: Definition of metadata for execution details

In this section, we define the final element of our provenance taxonomy: execution details (T6).

What: Executions are everything that is related to the analysis but which is not covered by the other categories. They constitute pure retrospective provenance: a record of what actually happened during the workflow run.

How: We distinguish four components in this category:

EX1 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 (SABDab). 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 Section 4.7, 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 (Section 4.5).

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.

4.9 Discussion of the provenance taxonomy

The provenance taxonomy defined in the previous sections was established on the basis of representative provenance questions for potential use cases of ROs produced by a workflow for epitope prediction.

In reality, the list of use cases of CWLProv ROs is likely longer than the few we considered for our example workflow. For example, there may be stakeholders which are interested in the funding of the research, which is an element which is currently not included in our provenance taxonomy. Nevertheless, we are convinced that the use cases we covered are realistic and applicable to more workflows than our example workflow, and that the provenance questions and taxonomy described here are a useful starting point which can be extended when other use cases are identified.

The first practical application is that our provenance questions are within the scope of the Competency Questions⁵ gathered by the RO-Crate Workflow Run profile working group. These competency questions will serve as a basis for the new RO-Crate Workflow Run profile. This way, our work not only applies to CWLProv ROs, but also to a much broader specification which can be adopted by a variety of workflow languages.

Secondly, our work identifies a use case (U5) where the RO in question is not generated for publication in companionship with a scientific article, but as standard output of a web-based tool. This scenario calls for a light-weight RO profile, in which some data entities are stored in an external repository

⁵<https://github.com/ResearchObject/workflow-run-crate/issues?q=is%3Aissue+is%3Aopen+label%3ARequirement>

instead of in the RO itself. Future work will need to explore the desired properties of these ROs further, as well as analyze the potential implications and risks of data exclusion from the RO, and translate this to metadata which is particularly important to include in these ROs.

5 ANALYSIS OF CWLPROV METADATA

In Section 4, we presented potential use cases for ROs associated with our example workflow and specified the metadata required to support them. In this section, our aim is to evaluate CWLProv provenance for the presence of the components of the provenance taxonomy, and in this way answer RQ2.

In contrast to the original CWLProv paper, our method explicitly evaluates CWLProv from a provenance analytics perspective (how ROs are used in practice). This analysis is conceptually challenging, because it requires us to discriminate between different types of representation (structured or unstructured, automatically or manually supplied).

5.1 Requirements

In this analysis, we consider for each of the provenance types and subtypes defined in Section 4 whether they adhere to the following requirements:

- R1** The provenance subtype is represented in the RO.
- R2** The representation of the subtype is in a structured format.

5.2 Methodology

We perform a qualitative analysis of CWLProv 0.6.0, based on ROs generated with the CWL reference implementation *cwltool*. We consider each of the 6 components defined in Section 4. We discriminate between structured representation in RDF (provenance graph and *manifest.json*), structured annotations in CWL-specific documents (*packed.cwl* and *primary-job.json*), and *unstructured* documents such as the execution log.

We consider 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 is *partially represented* if only a subset of the metadata is included in the RO, or it is not clear how to annotate this in the workflow or input object document.

5.3 Results

Here we connect the provenance as described in Section 4 to the requirements as defined in Section 5.1.

Table 2 summarizes the results of the analysis. Of all 6 provenance elements, **execution details** (T6) are best represented. The provenance graph contains start and end timestamps at both workflow and step granularity⁶ (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 included 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 mentions only the workflow and its steps. The steps are not further explained in the provenance graph, nor 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 poorly 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.

Although the underlying **software** (T3) 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 added 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 part of the RDF provenance graph, 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 represented. 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 only present in *primary-job.json*.

The **scientific context** (T1) can be partly represented via 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).

5.4 Discussion

Here, we presented the results of our analysis of the CWLProv 0.6.0 specification, based on the provenance taxonomy described in Section 4.

In this analysis, we distinguished between description in RDF, in CWL-specific documents (*packed.cwl* and *primary-job.json*), and in an unstructured format (e.g. the execution log).

⁶A side note to be made here is that in *cwltool*-generated ROs of executions which reused cached results, the timestamps correspond to the final execution, not the original

run which computed the results. We opened an issue about this: <https://github.com/common-workflow-language/cwltool/issues/1689>

Table 2: Summarized results of CWLProv 0.6.0 analysis. The table shows for each taxonomy component defined in Section 4 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* and *primary-job.json*), and unstructured representation (e.g. execution log).

Type	Subtype	Name	RDF	packed.cwl	primary-job.json	unstructured
T1	SC1	Workflow-level		(■)		
	SC2	Data-level			(□)	
	SC3	Execution-level *				
T2	D1	Data identification			(□)	
	D2	File characteristics	□		(□)	□
	D3	Data access			(□)	
	D4	Data 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	Timestamps	■			
	EX2	Consumed resources				□
	EX3	Workflow engine	□			
	EX4	Human agent	■			■

Firstly, we observe that very few categories are considered fully represented in provenance (R1). Except the workflow (T4), which is contained in its entirety in *packed.cwl*, all components are missing at least one element. The computational environment (T5) is almost entirely absent from provenance.

Secondly, the information that is contained in the RO is mostly restricted to CWL-specific files (*packed.cwl* and *primary-input.json*). Many of the annotations supplied in the workflow or input object are not transferred to the RDF provenance record, and can therefore not be extracted with SPARQL queries.

Finally, we observe that much of the required metadata for data (T2), software (T3, T4), and scientific context (T1) depends heavily on manual annotations supplied by the workflow author. Authors are encouraged to add this metadata⁷, but there are no formal specifications how to supply these annotations. However, if provenance is to be queried systematically, there is a need to issue stricter guidelines to do this, and in this way meet R2.

Our analysis was an evaluation of the CWLProv specification, not of the workflow engine which produced the RO. For this reason, we did not consider the amount of effort that was required from the workflow authors to supply this information. However, since much of the metadata was strongly dependent on manual input from the workflow authors, it is not reasonable to assume that this information will be present in all CWLProv ROs,

especially in complex workflows with many input parameters, steps, and software dependencies. In Section 7.6, we explore strategies to partially automate the inclusion of these metadata descriptions to reduce the burden on workflow authors.

6 DESIGN AND IMPLEMENTATION OF AN ANNOTATION SCHEME FOR INPUT DATA

In Section 5, we analyzed CWLProv provenance for the presence of the provenance types and subtypes defined in Section 4. Among other gaps, we observed that although some of this metadata can be provided via manual annotations of the workflow and input files, the CWL standards do not formally specify how this information should be included. In this section, we propose an annotation scheme which can enrich input data with semantic annotations and also enables authors to annotate the workflow run itself (i.e., the combination of a particular workflow and its configuration settings, SC3). In this way, we (partially) answer RQ3.

The remainder of this section is organized as follows. In Section 6.1, we define the requirements for a satisfactory standard. In Section 6.2, we describe the key ideas behind the annotation scheme. In Section 6.3, we describe how the current CWL standards (v1.2) support annotation of the input object document. Subsequently, in Section 6.4, we propose an extension of the annotation scheme which can be included in a future release of the CWL standards and provide examples of how it supports richer descriptions of

⁷https://www.commonwl.org/user_guide/17-metadata/index.html

workflow inputs. In Section 6.5, we use the annotation scheme to annotate the input object document of the epitope prediction workflow and analyze its strengths and limitations.

Our annotation scheme is grounded in real life use cases and designed to be easy for workflow authors to understand and use. Moreover, it is compatible with the existing Bioschemas specification, which it extends with additional terms to represent the history of processed workflow inputs.

6.1 Requirements

Based on the results of the analysis described in Section 5, the input annotation scheme should meet the following requirements:

- IR1** Represent the elements defined in **D1** and **D3**.
- 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 (**SC3**)

6.2 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 [18] initiative. Schema.org terms are also adopted by related efforts such as the RO-Crate specification [38].
- 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.

6.3 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:`.

6.3.1 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**).

6.3.2 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 v1.0⁸, in this way complying with **IP1**. Table 3 shows how the Schema.org terms relate to the required metadata specified in **D1** and **D3**.

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 Table 3, they may choose to extend this annotation scheme with domain-specific ontologies (such as EDAM [24]), 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**).

6.3.3 Annotations for a FAIR file. The following is an example of a standalone dataset with its own identifier. In addition to the CWL-specific *format* field (*line 4*), we provided additional Schema.org terms in order to comply with **IR1**. In principle, providing the identifier and version with a description (*lines 5-7*) is sufficient for unambiguous identification, since the identifier resolves to a landing page with additional information. However, for the purpose of our example, we added all other terms from Table 3 manually as well (*lines 8-16*).

```
1 FAIR_file:
2   class: File
3   location: path://path/to/6nzn.pdb
4   format: http://edamontology.org/format_1476 # pdb
5   s:identifier: https://doi.org/10.2210/pdb6nzn/pdb
6   s:version: "1.4"
7   s:description: "Amyloid fibril structure of
8     ↪ glucagon in pdb format."
9   s:name: "6NZN"
```

⁸<https://bioschemas.org/profiles/Dataset/1.0-RELEASE>

Table 3: Schema.org terms to use to express the metadata elements described in D1 and D3. 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

```

9   s:citation:
10    s:identifier:
11      ↪ https://doi.org/10.1038/s41594-019-0238-6
12  s:dateCreated: 2019-02-14
13  s:dateModified: 2019-12-18
14  s:includedInDataCatalog: # PDB
15    s:identifier:
16      ↪ https://doi.org/10.25504/FAIRsharing.2t35ja
17  s:distribution:
18    ↪ https://ftp.wwpdb.org/pub/pdb/data/structures
19    ↪ /divided/pdb/nz/pdb6nzn.ent.gz
20  s:license: https://spdx.org/licenses/CC0-1.0

```

Appendix B.1.1 shows an example of an annotated dataset which is not FAIR.

6.3.4 Adding domain-specific annotations. The CWL standards also support adding domain-specific ontologies. Here we add extra information about the biological interpretation of the data, using terms from the EDAM ontology (with *edam:* for <http://edamontology.org/>). In addition to the identifier, version and description (lines 5-7), we explicitly define this file as a dataset and protein structure (lines 8-10) and express the scientific domain to which it is related (line 11).

```

1  domain_annotations_file:
2    class: File
3    location: path://path/to/6nzn.pdb
4    format: http://edamontology.org/format_1476 # pdb
5    s:identifier: https://doi.org/10.2210/pdb6nzn/pdb
6    s:version: "1.4"
7    s:description: "Amyloid fibril structure of
8      ↪ glucagon in pdb format."
9    s:additionalType:
10      - s:Dataset
11      - edam:data_1460 # protein structure
12    edam:has_topic: edam:topic_2814 # protein
13      ↪ structure analysis

```

6.3.5 Annotations of a collection of input parameters. This example shows how scientific context can be represented, even for parameters which are not *Files* or *Directories*. Lines 1-5 denote the value of the workflow input parameters. The entire set of parameters is described via *s:description* (line 7), providing a mechanism to distinguish ROs of different workflow runs from each other.

```

1  input1: "string value"
2  input2: 4
3  input3:
4    class: File
5    location: path://path/to/file.txt
6
7  s:description: "Workflow run without input feature
8    ↪ X to test effect on model performance."

```

6.4 Proposed extension of CWL standards to support richer annotations.

The previous section explained how the latest release of the CWL standards can support metadata annotations. However, with the

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

presented annotation scheme, authors will find it difficult to explain the history of processed input data in a structured format (IR3). In addition, it is non-trivial to explain that a dataset (which can be a collection of files) is the result of a database query (IR4).

In this section, we propose an extension to the CWL standards which enables authors to annotate *Arrays* (IR2), and represent queries and processing operations which lead to the dataset they used as input for the workflow execution.

6.4.1 Format. We propose that authors represent the history of their datasets as a sequence of actions in the input object document. These actions are performed on an initial dataset and produce a result. The result of an action can be the input of another.

To avoid obfuscating the input object document, the metadata must be listed under a *cwlprov:prov* field, separate from the input values (IP3). This is analogous to the *overrides* field in the current CWL standards⁹.

6.4.2 Vocabulary. We used Schema.org terms to express search and processing actions. Table 4 lists the properties defined for *s:Actions* and (more specific) *s:SearchActions* and explains how they can be used in the annotation of CWL documents. In general, *Actions* are performed on an *object*, producing a *result*. The action can be initiated by an *agent*, using an *instrument*. *SearchActions* can be based on a *query*. The moment the action was performed can be represented via *endTime*.

6.4.3 Example annotation of a sequence of actions. In the following example, a database search was performed (lines 2-8), followed by a filtering operation (lines 10-15). The resulting dataset was used as an input for the workflow (lines 17-19). Both actions are listed under *cwlprov:prov* (line 1). The result of the search action (line 7) corresponds to the object of the filtering operation (line 12). In this example, the query is provided in a human-readable format, but the exact query which was issued (a JSON string) could also be used.

```

1  cwlprov:prov:
2    pdb_search:
3      s:additionalType: s:SearchAction
4      s:query: "All proteins with at least 2 chains
5        ↪ deposited between 2010 and 2022"
6      s:object:

```

⁹<https://github.com/common-workflow-language/cwltool/#overriding-workflow-requirements-at-load-time>

```

6       s:identifier: https://bio.tools/pdb
7       s:result: pdb_search_result
8       s:endTime: 2022-08-01
9
10      filtering_action:
11        s:additionalType: s:Action
12        s:object: pdb_search_result
13        s:instrument:
14          s:identifier: https://bio.tools/pisces
15          s:result: filtered_pdb_dataset
16
17      filtered_pdb_dataset:
18        class: Directory
19        location: path://path/to/directory/

```

In Appendix B.2, we provide an example of an input dataset which was merged from two initial datasets.

6.5 Analysis

6.5.1 Annotate input data for epitope prediction workflow. In Appendix B.3, we show how we would apply the annotation scheme to the inputs of the epitope prediction workflow. Because of the annotation scheme proposed in this section, **D1** and **D3** are now represented in a structured format (**R2**), as well as **SC3**.

6.6 Discussion

In Section 6, we proposed an annotation scheme with which CWL workflow authors can enrich the input data they use with structured metadata expressing their origin. In addition, the model facilitates annotations related to specific executions of the workflow with a particular input configuration.

Firstly, it should be mentioned that the proposed annotations are limited to inputs of the workflow. Data entities which are intermediate or final outputs are not enriched with metadata via annotations in the input object document. A strategy via which this information could be supplied could involve adding similar annotations to *Workflow* and *CommandLineTool* output parameters. Workflow systems could also support annotations of data after execution of the workflow, to facilitate representing interpretation of the workflow results. In the case where a workflow is built incrementally, these descriptions could convey the motivation behind the addition of new workflow steps (e.g. examination of the results of the first step could motivate the inclusion of an extra preprocessing operation).

In addition, we did not consider the structured annotation of CWL tool and workflow descriptions (which are software entities). Future work should focus on how to apply the Bioschemas *ComputationalTool*¹⁰ and *ComputationalWorkflow*¹¹ profiles to CWL *Workflow* and *CommandLineTool* descriptions in order to improve the representation of **WF1**.

The *CommandLineTool* documents could also be used to annotate the underlying command-line program. This is especially important when the tool runs a custom script which can not be represented under *SoftwareRequirements*. We propose that these annotations

are nested under the field *s:mainEntity*, to clearly separate them from the annotations describing the *CommandLineTool* itself.

Finally, we limited our annotation scheme to Schema.org vocabulary. Although other, domain-specific ontologies are supported, we considered specific guidelines for their application in CWL out of scope for the CWL standards. With the proposed Schema.org terms, authors can represent metadata which was part of our provenance taxonomy (**D1** and **D3**), and generalizable to most workflows. If necessary, additional guidelines can be established within the respective scientific communities for the use of particular domain-specific ontologies in CWL.

7 DESIGN OF AN EXTENSION OF THE CWLPROV PROVENANCE GRAPH

In Section 5, we analyzed CWLProv for the representation of the provenance taxonomy we defined in Section 4. 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 Section 2.2, 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 **RQ3**.

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 Section 7.1, we first describe the requirements and principles which we used in this design. Subsequently, in Section 7.2, 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 (Section 7.3). We describe how we have partially realized this design in *cwltool* (Section 7.4). Finally, we perform a conceptual analysis in Section 7.5 and discuss the design in Section 7.6.

Our analysis of CWLProv revealed incomplete RDF description of the workflow execution. Here, we design a solution, compatible with the annotation scheme for input data and existing metadata fields in the CWL standards, and considering the initial design of the provenance graph.

7.1 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 Table 1.

¹⁰<https://bioschemas.org/profiles/ComputationalTool/1.0-RELEASE>

¹¹<https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE>

- PR2** Represent the annotations for *Files* and *Directories* proposed in Section 6.3.
- PR3** Represent the annotations for a collection of input values proposed in Section 6.3.
- PR4** Represent the annotations for *Actions* proposed in Section 6.4.

7.2 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 Section 7.1.

7.2.1 Representation of CWL metadata fields. Figure 5 shows how we extended the provenance graph to represent all current metadata fields (**PR1**). **In the new graph, every workflow component in Table 1 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.

7.2.2 Representation of input data annotations. This structure depicted in Figure 5 also supports representation of the input data annotations described in Section 6.3 (**PR2**), by transferring them to the data entities they describe (**G5**).

7.2.3 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**).

7.2.4 Representation of input data history. In the annotation scheme described in Section 6.4, the history of (processed) input data is expressed through *Actions*. Figure 6 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 Table 4. They are connected to input data entities (**G5**) via *s:result*.

7.3 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>
- **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).

7.4 Partial realization 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 Section 6.3.3, are propagated to the RDF provenance record. However, annotations of the collection of input parameter values (Section 6.3.5) or annotations under *cwlprov:prov* (Section 6.4), are currently not represented in RDF.

7.5 Conceptual analysis of the design extension

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.

7.5.1 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.

Q1

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

SELECT ?doi
WHERE {
    ?cit s:identifier ?doi .
    ?name s:citation ?cit .
}
```

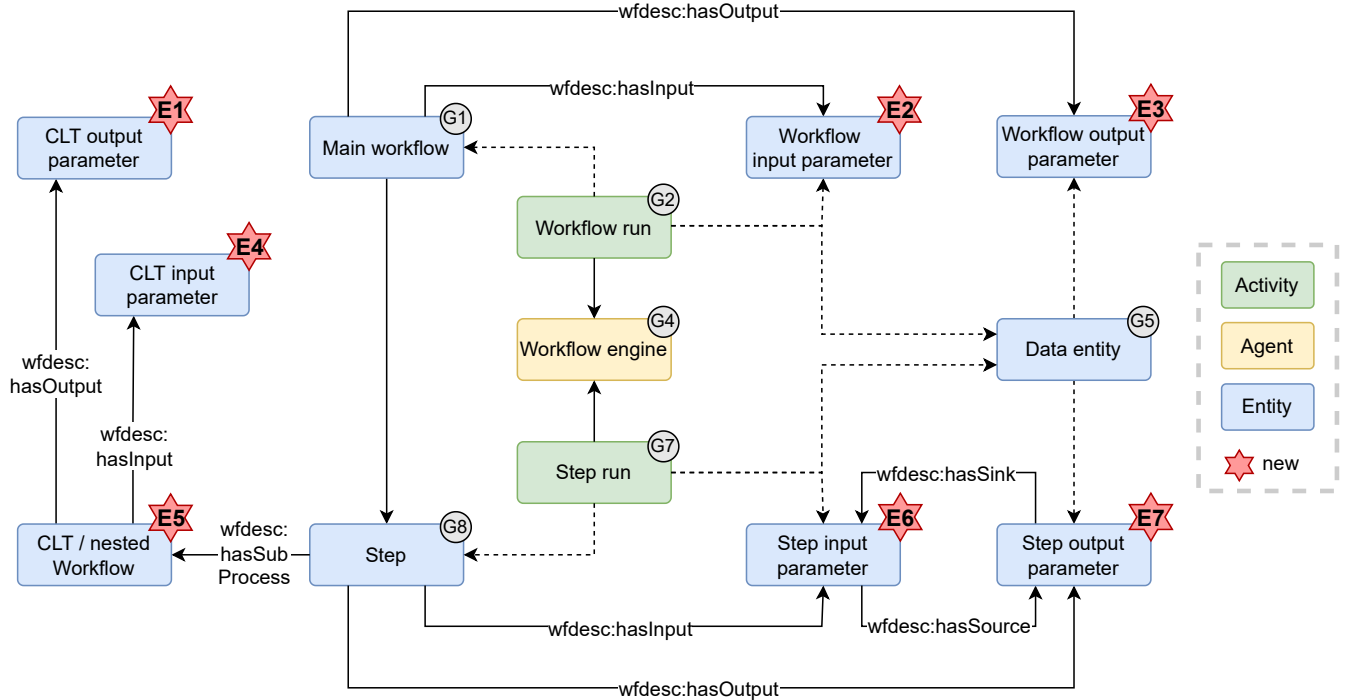


Figure 5: The RDF provenance graph depicted in Figure 3, now extended with *CommandLineTools* and parameter entities. Red stars mark nodes which are part of the design extension. Parameters are linked to their *Workflow*, *CommandLineTool* or step via *wfdesc:hasInput* and *wfdesc:hasOutput*. Node G5 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*.

Q2

```
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 .
}
```

7.5.2 *Which provenance types have improved:* As a result of the design presented in this section, we find that the representation of **WF1** and **WF2** has improved, since these provenance elements are now described in RDF with their associated metadata. In addition, the provenance subtypes represented in the input annotation scheme described in Section 6 (**D1**, **D3**, and **SC3**) are now included in RDF as well.

7.6 Discussion

In Section 7, we outlined an extension of the provenance graph such that it includes more of the provenance subtypes defined in

Section 4. Here, we discuss the implications and future directions of this work.

Firstly, although the provenance graph included the (container) environment as a separate entity, our design did not specify how the properties of this environment should be expressed. As far as we are aware, Schema.org currently does not include a formalized profile for Computational Environment. Until such a specification exists, CWLProv could represent the environment with a custom vocabulary using the *cwlprov* namespace as a temporary solution.

Secondly, we were only concerned with provenance representation, not with its acquisition. The collection of the meta-data may be different for each workflow system, which is why we considered it out of scope for our thesis.

Our analysis revealed that much of the required metadata needs to be added manually, and in Section 5.4 we stated that this can lead to incomplete provenance. Although manual annotations are difficult to replace in some cases (such as scientific context, **T1**), part of metadata collection can be automated to lift the burden on workflow authors.

For example, workflow systems could extract all relevant meta-data associated with a given entity from a data or software registry, if the workflow author supplied the identifier (or if the workflow system itself has stored its available tools with structured annotations). Workflow systems could support widely used registries

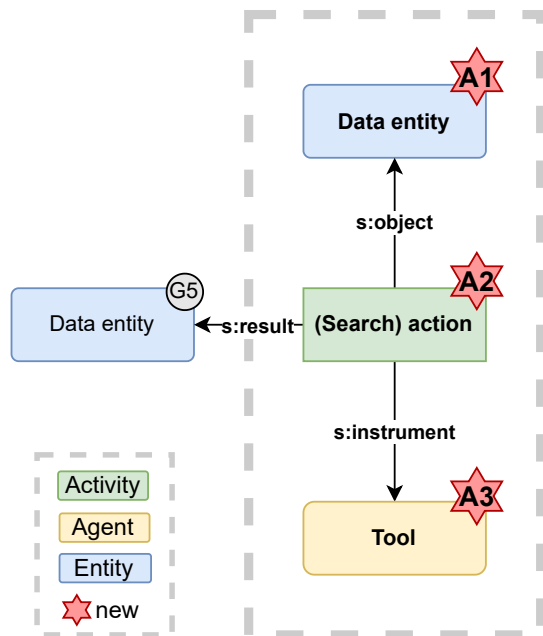


Figure 6: Actions represented in RDF provenance graph. Workflow input datasets (G5) are connected to the Actions (A2) which produced them via *s:result*. In addition, Actions can have properties such as *object* (A1) and *instrument* (A3).

like Zenodo¹² [14], BioTools [23], and WorkflowHub¹³, as well as standards like FAIRsharing.org.

In addition, file characteristics like last modification date and size (D2) can be retrieved from the filesystem on which the data is stored. In *cwltool*, we realized preservation of the last modification date of a remote file¹⁴. A next step could be to propagate this information to *manifest.json*.

Other information can be automatically extracted following a similar strategy, if a database returns metadata with the data during a download. An example of this is UniProt [44], which includes the database version and download date in the HTTP headers of the response¹⁵. Wide adoption of this strategy would require a common format in which databases supply their metadata.

Lastly, metadata can be extracted from container image labels, for which a standard format is specified by the Open Containers Initiative¹⁶. Application of this format to represent the metadata specified in ENV3 is within the scope of the BioContainers initiative [11]¹⁷.

Finally, before concluding this section, we discuss four methods via which the information contained in CWLProv

ROs can be applied for better transparency and reproducibility of workflow executions.

First of all, the extended provenance graph can be queried with SPARQL queries. The functionality of *cwlprov-py* could be extended with a set of SPARQL queries which are relevant to many workflows, and also allow users to issue their own SPARQL queries.

Secondly, workflow systems and repositories can use the structured information stored in the RO to make visualizations of the workflow and its execution. The command and parameter settings used in each step of the workflow could be represented in a table. These visualizations and data summaries can be used by workflow authors to make their analyses easier to understand in a scientific article (U2).

The provenance graph could also be exploited for automated generation of textual descriptions of the workflow execution, based on previous work on *data narratives* [16].

Future work should also focus on devising strategies for the annotation of data which is extracted from another RO. U5 exemplifies how workflow outputs (a trained model) can be reused as the input of another workflow (the trained model as a web service). FRESH [15] is a model for representation of the provenance of entities which are (intermediate) data products of a workflow run.

8 CONCLUSION

In a workflow-centric ecosystem of computational resources, ROs have been proposed as a means to represent the results of computational analyses in a structured format. In this section, we summarize how our work has answered the research questions defined in the Introduction.

For this work, we integrated methods from both Bioinformatics and Computer Science to arrive at a standard for metadata in ROs.

Firstly, we answered **RQ1** by defining 5 use cases of ROs associated with an example Bioinformatics workflow and establishing a provenance taxonomy based on questions associated with each use case.

Subsequently, to answer **RQ2**, we assessed the representation of each of the components of the provenance taxonomy in CWLProv 0.6.0, discriminating between RDF, structured but CWL-specific, and unstructured representation. We observed that computational environment was largely absent from the provenance record, that much of the required metadata was dependent on manual annotations, and that the RDF description of the workflow execution was incomplete.

In order to improve the provenance contained in CWLProv ROs (**RQ3**), we designed an annotation scheme with which workflow authors can add structured annotations to their input data. This scheme uses Schema.org vocabulary to express required metadata according to our provenance taxonomy, and can be extended with domain-specific ontologies. We demonstrated the application of our annotation scheme to the input document of our example Bioinformatics workflow.

Finally, we designed an extension to the RDF graph in order to support RDF representation of CWL-specific metadata fields and the structured annotations defined in our annotation scheme. We partially realized propagation of these annotations to RDF in the

¹²<https://zenodo.org/>

¹³<https://workflowhub.eu>

¹⁴<https://github.com/common-workflow-language/cwltool/pull/1676>

¹⁵https://www.uniprot.org/help/api_downloading






¹⁶<https://github.com/opencontainers/image-spec/blob/6ad7100eb087e43398e9ea8fe44fffc1501b8984/annotations.md>

¹⁷See the issue we opened about this topic here: <https://github.com/BioContainers/specs/issues/105>

CWL reference engine *cwltool* and demonstrated exploitation of the added metadata with SPARQL queries.

Summarizing, we established a model for provenance which will not only inform future releases of CWLProv, but also related standards like the RO-Crate Workflow Run profile. It also demonstrates that even considering only one example workflow can produce many new insights into the improvement of workflow reproducibility. Future directions of this work include its full realization in *cwltool*, an extension of the functionality of *cwlprov-py* to support structured provenance queries, and the specification of a model for the representation of metadata for RO output data entities which are used as input for another workflow. We are confident that the work presented in this thesis can serve as the basis for new, automated approaches to improve the transparency of computational analyses. Ultimately, this brings many scientific domains one step closer to a future in which computational reproducibility is not an exception, but the norm.

9 ACKNOWLEDGEMENTS

Alexandru Iosup  supervised the project. Michael R. Crusoe  was daily supervisor. The use cases arose from discussions with Katharina Waury  and other members from VU Bioinformatics Group, who conceptualized the epitope prediction workflow. Simone Leo , Stian Soiland-Reyes  and the rest of the RO-Crate Workflow Run Profile Working Group provided helpful feedback on the input data annotation scheme and extension of the provenance graph.

10 DATA AND CODE AVAILABILITY

The CWL implementation of the epitope prediction workflow can be found at <https://github.com/RenskeW/cwl-epitope>. Examples of the annotation scheme, provenance graph extension, and SPARQL queries are available at <https://github.com/RenskeW/cwlprov-provenance>. Code for the realization of input data propagation to RDF can be found in this pull request: <https://github.com/common-workflow-language/cwltool/pull/1678>. Code and data are also archived at <https://doi.org/10.5281/zenodo.7014949>.

REFERENCES

- [1] Georgios Andreadis, Alexandros Iosup, and Vincent van Beek. 2020. *Capelin: Fast Data-Driven Capacity Planning for Cloud Datacenters*. Master's thesis. <https://repository.tudelft.nl/islandora/object/uuid:d6d50861-86a3-4dd3-a13f-42d84db7af66/datastream/OBJ/>
- [2] D. J. Barlow, M. S. Edwards, and J. M. Thornton. 1986. Continuous and Discontinuous Protein Antigenic Determinants. *Nature* 322, 6081 (Aug. 1986), 747–748. <https://doi.org/10.1038/322747a0>
- [3] Khalid Belhajjame, Jun Zhao, Daniel Garijo, Matthew Gamble, Kristina Hettne, Raul Palma, Eleni Mina, Oscar Corcho, José Manuel Gómez-Pérez, Sean Bechhofer, Graham Klyne, and Carole Goble. 2015. Using a Suite of Ontologies for Preserving Workflow-Centric Research Objects. *Journal of Web Semantics* 32 (May 2015), 16–42. <https://doi.org/10.1016/j.websem.2015.01.003>
- [4] Khalid Belhajjame, Jun Zhao, Daniel Garijo, Kristina Hettne, Raul Palma, Óscar Corcho, José-Manuel Gómez-Pérez, Sean Bechhofer, Graham Klyne, and Carole Goble. 2014. The Research Object Suite of Ontologies: Sharing and Exchanging Research Data and Methods on the Open Web. *arXiv:1401.4307 [cs]* (Feb. 2014). [arXiv:1401.4307 \[cs\]](https://arxiv.org/abs/1401.4307)
- [5] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T.N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. 2000. The Protein Data Bank. *Nucleic Acids Research* 28, 1 (Jan. 2000), 235–242. <https://doi.org/10.1093/nar/28.1.235>
- [6] Henriette Capel, K. Anton Feenstra, and Sanne Abeln. 2022. *Multi-Task Learning to Leverage Partially Annotated Data for PPI Interface Prediction*. Preprint. In Review. <https://doi.org/10.21203/rs.3.rs-1269779/v1>
- [7] Christian Collberg, Todd Proebsting, Gina Moraila, Akash Shankaran, Zuoming Shi, and Alex M. Warren. 2013. Measuring Reproducibility in Computer Systems Research. (2013).
- [8] 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, Board on Mathematical Sciences and Analytics, Committee on Applied and Theoretical Statistics, Division on Engineering and Physical Sciences, Board on Research Data and Information, Committee on Science, Engineering, Medicine, and Public Policy, Policy and Global Affairs, and National Academies of Sciences, Engineering, and Medicine. 2019. *Reproducibility and Replicability in Science*. National Academies Press, Washington, D.C. 25303 pages. <https://doi.org/10.17226/25303>
- [9] Michael R. Crusoe, Sanne Abeln, Alexandros Iosup, Peter Amstutz, John Chilton, Nebojša Tijanić, Hervé Ménager, Stian Soiland-Reyes, Bogdan Gavrilović, Carole Goble, and The CWL Community. 2022. Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language. *Commun. ACM* 65, 6 (June 2022), 54–63. <https://doi.org/10.1145/3486897>
- [10] Richard Cyganiak, David Wood, Markus Lanthaler, Graham Klyne, Jeremy J. Carroll, and Brian McBride. 2014. RDF 1.1 Concepts and Abstract Syntax. <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.
- [11] Felipe da Veiga Leprevost, Björn A. Grüning, Saulo Alves Aflitos, Hannes L. Röst, Julian Uszkoreit, Harald Barsnes, Marc Vaudel, Pablo Moreno, Laurent Gatto, Jonas Weber, Mingze Bai, Rafael C. Jimenez, Timo Sachsenberg, Julianus Pfeuffer, Roberto Vera Alvarez, Johannes Griss, Alexey I. Nesvizhskii, and Yasset Perez-Riverol. 2017. BioContainers: An Open-Source and Community-Driven Framework for Software Standardization. *Bioinformatics* 33, 16 (Aug. 2017), 2580–2582. <https://doi.org/10.1093/bioinformatics/btx192>
- [12] Data Citation Synthesis Group. 2014. Joint Declaration of Data Citation Principles. *San Diego CA: FORCE11* (2014). <https://doi.org/10.25490/a97f-egyik>
- [13] James Dunbar, Konrad Krawczyk, Jinwoo Leem, Terry Baker, Angelika Fuchs, Guy Georges, Jiye Shi, and Charlotte M. Deane. 2014. SABDab: The Structural Antibody Database. *Nucleic Acids Research* 42, D1 (Jan. 2014), D1140–D1146. <https://doi.org/10.1093/nar/gkt1043>
- [14] European Organization For Nuclear Research and OpenAIRE. 2013. Zenodo. <https://doi.org/10.25495/7GXX-RD71>
- [15] Alban Gaignard, Hala Skaf-Molli, and Khalid Belhajjame. 2020. Findable and Reusable Workflow Data Products: A Genomic Workflow Case Study. *Semantic Web* 11, 5 (Aug. 2020), 751–763. <https://doi.org/10.3233/SW-200374>
- [16] Yolanda Gil and Daniel Garijo. 2017. Towards Automating Data Narratives. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, Limassol Cyprus, 565–576. <https://doi.org/10.1145/3025171.3025193>
- [17] Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R. Crusoe, Kristian Peters, and Daniel Schober. 2020. FAIR Computational Workflows. *Data Intelligence* 2, 1-2 (Jan. 2020), 108–121. https://doi.org/10.1162/dint_a_00033
- [18] Alasdair J G Gray, Carole Goble, and Rafael C Jimenez. 2017. From Potato Salad to Protein Annotation. (Oct. 2017), 5.
- [19] Michael R. Gryk and Bertram Ludäscher. 2017. Workflows and Provenance: Toward Information Science Solutions for the Natural Sciences. *Library Trends* 65, 4 (2017), 555–562. <https://doi.org/10.1353/lib.2017.0018>
- [20] R V Guha, Dan Brickley, and Steve Macbeth. 2015. Big Data Makes Common Schemas Even More Necessary. (2015), 28.
- [21] Qingzhen Hou, Bas Stringer, Katharina Waurly, Henriette Capel, Reza Haydarlou, Fuzhong Xue, Sanne Abeln, Jaap Heringa, and K Anton Feenstra. 2021. SeRenDIP-CE: Sequence-Based Interface Prediction for Conformational Epitopes. *Bioinformatics* 37, 20 (Oct. 2021), 3421–3427. <https://doi.org/10.1093/bioinformatics/btab321>
- [22] Alexandros Iosup, Laurens Versluis, Animesh Trivedi, Erwin van Eyk, Lucian Toader, Vincent van Beek, Giulia Frascaria, Ahmed Musafir, and Sacheendra Talluri. 2019. The AtLarge Vision on the Design of Distributed Systems and Ecosystems. *arXiv:1902.05416 [cs]* (Feb. 2019). [arXiv:1902.05416 \[cs\]](https://arxiv.org/abs/1902.05416)
- [23] Jon Ison, Hans Ienasescu, Piotr Chmura, Emil Rydza, Hervé Ménager, Matúš Kalaš, Veit Schwämmle, Björn Grüning, Niall Beard, Rodrigo Lopez, Severine Duvaud, Heinz Stockinger, Bengt Persson, Radka Svobodová Vařeková, Tomáš Raček, Jiří Vondrášek, Hedi Peterson, Ahto Salumets, Inge Jonassen, Rob Hooft, Tommi Nyrrönen, Alfonso Valencia, Salvador Capella, Josep Gelpi, Federico Zambelli, Babis Savakis, Brane Leskošek, Kristoffer Rapacki, Christophe Blanchet, Rafael Jimenez, Arlindo Oliveira, Gert Vriend, Olivier Collin, Jacques van Helden, Peter Løngreen, and Søren Brunak. 2019. The Bio.Tools Registry of Software Tools and Data Resources for the Life Sciences. *Genome Biology* 20, 1 (Dec. 2019), 164. <https://doi.org/10.1186/s13059-019-1772-6>
- [24] J. Ison, M. Kalas, I. Jonassen, D. Bolser, M. Uludag, H. McWilliam, J. Malone, R. Lopez, S. Pettifer, and P. Rice. 2013. EDAM: An Ontology of Bioinformatics Operations, Types of Data and Identifiers, Topics and Formats. *Bioinformatics* 29, 10 (May 2013), 1325–1332. <https://doi.org/10.1093/bioinformatics/btt113>
- [25] Wolfgang Kabsch and Christian Sander. 1983. Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers* 22, 12 (Dec. 1983), 2577–2637. <https://doi.org/10.1002/bip.360221211>
- [26] Gregg Kellogg, Pierre-Antoine Champin, Dave Longley, Manu Sporny, Markus Lanthaler, and Niklas Lindström. 2020. JSON-LD 1.1: A JSON-based Serialization for Linked Data. <https://www.w3.org/TR/json-ld11/>.
- [27] Fahim Halim Khan. 2014. Chapter 25 - Antibodies and Their Applications. In *Animal Biotechnology*. Academic Press, San Diego, 473–490.
- [28] Farah Zaib Khan, Stian Soiland-Reyes, Richard O Sinnott, Andrew Lonie, Carole Goble, and Michael R Crusoe. 2019. Sharing Interoperable Workflow Provenance: A Review of Best Practices and Their Practical Application in CWLProv. *GigaScience* 8, 11 (Nov. 2019), giz095. <https://doi.org/10.1093/gigascience/giz095>
- [29] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific Containers for Mobility of Compute. *PLOS ONE* 12, 5 (May 2017), e0177459. <https://doi.org/10.1371/journal.pone.0177459>
- [30] Franck Michel and The Bioschemas Community. 2018. Bioschemas & Schema.Org: A Lightweight Semantic Layer for Life Sciences Websites. *Biodiversity Information Science and Standards* 2 (May 2018), e25836. <https://doi.org/10.3897/biss.2.25836>
- [31] Luc Moreau, Paolo Missier, Khalid Belhajjame, Reza B'Far, James Cheney, iMinds Coppens, Stephen Cresswell, Yolanda Gil, Paul Groth, Graham Klyne, Timothy Lebo, Jim McCusker, Simon Miles, James Myers, Satya Sahoo, and Curt Tilmes. 2013. PROV-DM: The PROV Data Model. <https://www.w3.org/TR/2013/REC-prov-dm-20130430/>.
- [32] Beatriz Pérez, Julio Rubio, and Carlos Sáenz-Adán. 2018. A Systematic Review of Provenance Systems. *Knowledge and Information Systems* 57, 3 (Dec. 2018), 495–543. <https://doi.org/10.1007/s10115-018-1164-3>
- [33] Michael Remmert, Andreas Biegert, Andreas Hauser, and Johannes Söding. 2012. HHblits: Lightning-Fast Iterative Protein Sequence Searching by HMM-HMM Alignment. *Nature Methods* 9, 2 (Feb. 2012), 173–175. <https://doi.org/10.1038/nmeth.1818>
- [34] Inbal Sela-Culang, Vered Kunik, and Yanay Ofran. 2013. The Structural Basis of Antibody-Antigen Recognition. *Frontiers in Immunology* 4 (2013). <https://doi.org/10.3389/fimmu.2013.00302>
- [35] Arfon M. Smith, Daniel S. Katz, Kyle E. Niemeyer, and FORCE11 Software Citation Working Group. 2016. Software Citation Principles. *PeerJ Computer Science* 2 (Sept. 2016), e86. <https://doi.org/10.7717/peerj-cs.86>
- [36] Stian Soiland-Reyes, Sean Bechhofer, Oscar Corcho, Khalid Belhajjame, Daniel Garijo, Esteban García Cuesta, and Raul Palma. 2016. The Wfdesc Ontology. <https://wf4ever.github.io/ro/2016-01-28/wfdesc/>.
- [37] Stian Soiland-Reyes, Sean Bechhofer, Oscar Corcho, Khalid Belhajjame, Daniel Garijo, Esteban García Cuesta, and Raul Palma. 2016. The Wfprov Ontology. <https://wf4ever.github.io/ro/2016-01-28/wfprov/>.
- [38] Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José M. Fernández, Daniel Garijo, Björn Grüning, Marco La Rosa, Simone Leo, Eoghan Ó Carragáin, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, and Carole Goble. 2022. Packaging Research Artefacts with RO-Crate. *Data Science* (Jan. 2022), 1–42. <https://doi.org/10.3233/DS-210053>
- [39] Joan Starr, Eleni Castro, Mercè Crosas, Michel Dumontier, Robert R. Downs, Ruth Duerr, Laurel L. Haak, Melissa Haendel, Ivan Herman, Simon Hodson, Joe Hourclé, John Ernest Kratz, Jennifer Lin, Lars Holm Nielsen, Amy Nurnberger, Stefan Proell, Andreas Rauber, Simone Sacchi, Arthur Smith, Mike Taylor, and

- Tim Clark. 2015. Achieving Human and Machine Accessibility of Cited Data in Scholarly Publications. *PeerJ Computer Science* 1 (May 2015), e1. <https://doi.org/10.7717/peerj-cs.1>
- [40] Victoria Stodden, Matthew S. Krafczyk, and Adhithya Bhaskar. 2018. Enabling the Verification of Computational Results: An Empirical Evaluation of Computational Reproducibility. In *Proceedings of the First International Workshop on Practical Reproducible Evaluation of Computer Systems*. ACM, Tempe AZ USA, 1–5. <https://doi.org/10.1145/3214239.3214242>
- [41] Victoria Stodden, Marcia McNutt, David H. Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A. Heroux, John P.A. Ioannidis, and Michela Taufer. 2016. Enhancing Reproducibility for Computational Methods. *Science* 354, 6317 (Dec. 2016), 1240–1241. <https://doi.org/10.1126/science.aah6168>
- [42] Victoria Stodden, Jennifer Seiler, and Zhaokun Ma. 2018. An Empirical Analysis of Journal Policy Effectiveness for Computational Reproducibility. *Proceedings of the National Academy of Sciences* 115, 11 (March 2018), 2584–2589. <https://doi.org/10.1073/pnas.1708290115>
- [43] Bas Stringer, Hans de Ferrante, Sanne Abeln, Jaap Heringa, K Anton Feenstra, and Reza Haydarlou. 2022. PIPENN: Protein Interface Prediction from Sequence with an Ensemble of Neural Nets. *Bioinformatics* 38, 8 (April 2022), 2111–2118. <https://doi.org/10.1093/bioinformatics/btac071>
- [44] The UniProt Consortium, Alex Bateman, Maria-Jesus Martin, Sandra Orchard, Michele Magrane, Rahat Agivetova, Shadab Ahmad, Emanuele Alpi, Emily H Bowler-Barnett, Ramona Britto, Borisas Bursteinas, Hema Bye-A-Jee, Ray Coetzee, Austra Cukura, Alan Da Silva, Paul Denny, Tunca Dogan, ThankGod Ebenezer, Jun Fan, Leyla Garcia Castro, Penelope Garmiri, George Georghiou, Leonardo Gonzales, Emma Hatton-Ellis, Abdulrahman Hussein, Alexandr Ignatchenko, Giuseppe Insana, Rizwan Ishtiaq, Petteri Jokinen, Vishal Joshi, Dushyanth Jyothi, Antonia Lock, Rodrigo Lopez, Aurelien Luciani, Jie Luo, Yvonne Lussi, Alistair MacDougall, Fabio Madeira, Mahdi Mahmoudy, Manuela Menchi, Alok Mishra, Katie Moulang, Andrew Nightingale, Carla Susana Oliveira, Sangya Pundir, Guoying Qi, Shriya Raj, Daniel Rice, Milagros Rodriguez Lopez, Rabie Saidi, Joseph Sampson, Tony Sawford, Elena Speretta, Edward Turner, Nidhi Tyagi, Preethi Vasudev, Vladimir Volynkin, Kate Warner, Xavier Watkins, Rossana Zaru, Hermann Zellner, Alan Bridge, Sylvain Poux, Nicole Redaschi, Lucila Aimo, Ghislaine Argoud-Puy, Andrea Auchincloss, Kristian Axelsen, Parit Bansal, Delphine Baratin, Marie-Claude Blatter, Jerven Bolleman, Emmanuel Boutet, Lionel Breuza, Cristina Casals-Casas, Edouard de Castro, Kamal Chikh Echouk, Elisabeth Coudert, Beatrice Cuche, Mikael Doche, Dolnide Dornevil, Anne Estreicher, Maria Livia Famiglietti, Marc Feuermann, Elisabeth Gasteiger, Sebastien Gehant, Vivienne Gerritsen, Arnaud Gos, Nadine Gruaz-Gumowski, Ursula Hinz, Chantal Hulo, Nevila Hyka-Nouspikel, Florence Jungo, Guillaume Keller, Arnaud Kerhornou, Vicente Lara, Philippe Le Mercier, Damien Lieberherr, Thierry Lombardot, Xavier Martin, Patrick Masson, Anne Morgat, Teresa Batista Neto, Salvo Paesano, Ivo Pedruzzi, Sandrine Pilbout, Lucille Poursel, Monica Pozzato, Manuela Pruess, Catherine Rivoire, Christian Sigrist, Karin Sonesson, Andre Stutz, Shyamala Sundaram, Michael Tognolli, Laure Verbregue, Cathy H Wu, Cecilia N Arighi, Leslie Arminski, Chuming Chen, Yongxing Chen, John S Garavelli, Hongzhan Huang, Kati Laiho, Peter McGarvey, Darren A Natale, Karen Ross, C R Vinayaka, Qinghua Wang, Yuqi Wang, Lai-Su Yeh, Jian Zhang, Patrick Ruch, and Douglas Teodoro. 2021. UniProt: The Universal Protein Knowledgebase in 2021. *Nucleic Acids Research* 49, D1 (Jan. 2021), D480–D489. <https://doi.org/10.1093/nar/gkaa1100>
- [45] The W3C SPARQL Working Group. 2013. SPARQL 1.1 Overview. <http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/>.
- [46] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR Guiding Principles for Scientific Data Management and Stewardship. *Scientific Data* 3, 1 (Dec. 2016), 160018. <https://doi.org/10.1038/sdata.2016.18>
- [47] Gang Xu, Qinghua Wang, and Jianpeng Ma. 2020. OPUS-TASS: A Protein Backbone Torsion Angles and Secondary Structure Predictor Based on Ensemble Neural Networks. *Bioinformatics* 36, 20 (Dec. 2020), 5021–5026. <https://doi.org/10.1093/bioinformatics/btaa629>

A PROVENANCE QUESTIONS

This section presents representative questions associated to each of the use cases defined in Section 4.1. Here, we relate them to the components of the provenance taxonomy defined in Section 4.

U1: Workflow development

- (1) What is the influence of a different model architecture on performance? **SW1, SW2, WF1, WF2**
- (2) What is the influence of removing input feature X on performance? **WF1, SC2**
- (3) What is the influence of including filtering step X on performance? **WF1, SC2**
- (4) What is the influence of ... on training time? **EX1**
- (5) What is the influence of a different training set on performance? **D1**

A.1 U2: Publishing the workflow

- (1) What are all the resources which contributed to this research (and should be cited)? **D1, SW1, WF1, ENV3**
- (2) What are all the input data I used? **D4**
- (3) Which software was used in this workflow? **SW1**
- (4) I reused an existing CWL *CommandLineTool* description. How do I give credit to the original authors? **WF1**
- (5) I wrote a CWL *CommandLineTool* description for existing software. How do I cite it? **SW1**
- (6) I reused a custom, unpublished script made by one of my collaborators. How do I give credit to the original authors? **SW1**
- (7) I wrote a custom script based on existing code. How do I give credit to the original authors? **SW1**
- (8) I reused a Dockerfile or Docker container which was written or built by someone else. How do I give credit to the original authors? **ENV3**
- (9) Which of my colleagues contributed to this workflow to whom I should give credit and/or propose as co-authors? What were their contributions? **WF1, EX4**
- (10) From which workflow(s) was this workflow derived? **WF1**

A.2 U3: Understanding the workflow

- (1) What is the goal of this analysis? What was the hypothesis of this experiment? **SC3**
- (2) Why was this step included? **SC1**
- (3) Why was this combination of steps chosen? **SC1**
- (4) Why was this set of values chosen as inputs for this workflow execution? **SC3**
- (5) How was this figure from the paper generated? **D4**
- (6) Why were these input settings chosen? **SC3**
- (7) What is the interpretation of this (intermediate) output? **SC2**
- (8) Which input features were used for the model? **WF1**
- (9) Which related tasks were predicted by the model? **WF1**
- (10) What are the values of PSP19 for each amino acid type? **SC2**
- (11) What was the performance of the model? **SC2**
- (12) What was model architecture? **SW1, SW2**

- (13) Where can I find more information about BioDL dataset? **D1**
- (14) How were UniProt IDs mapped to PDB IDs? **WF1**
- (15) Which SABDab query was used to generate the summary file? **SC2, D1**
- (16) Which query was issued to PDB? **SC2, D1**
- (17) Which UniProt IDs are part of BioDL? **D1**

A.3 U4: Reproducing the workflow

- (1) Which HHBlits reference database was used? Which version? **D1**
- (2) Where can I download HHBlits reference database? (It was not stored in the RO because of its large size.) **D3**
- (3) Which version of HHBlits was used? **SW1, ENV1**
- (4) Which software (and their versions) was installed on the system? **ENV1**
- (5) I pulled a Docker image. Is this the same as which was used in the original analysis? **ENV3**
- (6) Original author used Dockerfile. Is the image I built on a rerun the same as the one in the original analysis? **ENV3**
- (7) What are resource requirements as specified in the workflow description? **WF3**
- (8) How many CPUs are required for this step? **WF3, EX2**
- (9) How much memory is necessary for this step? **WF3, EX2**
- (10) How much memory was used in the original run? **EX2**
- (11) Does this step need network access? **WF3, EX2**
- (12) Which CPU/GPU was installed? **ENV2**
- (13) How long does this step take? **WF3, EX1**
- (14) What was the last modification date of this input file? **D2**
- (15) Which Python version was used in this step? **SW1, ENV1**
- (16) When was PDB queried? **EX1**
- (17) When was SABDab queried? **EX1**
- (18) Which format does the PDB batch download script need? **WF2**
- (19) The CWL *CommandLineTool* description did not describe all parameters of this command-line program. Which values were used for the other parameters? **WF2**

A.4 U5: Model as a web service

- (1) Which protein sequences were in the training set? **D1**
- (2) Which proteins had epitope annotations? **D1**
- (3) For which proteins did I make epitope predictions? **D1**
- (4) How should I cite this tool? **WF1**

B INPUT ANNOTATION EXAMPLES

B.1 Annotations supported by CWL Standards v1.2

B.1.1 Annotations for a non-FAIR file. When the dataset is not FAIR, the download or modification date (*line 7*) can serve as an alternative for version. The URL of the remote file is used as an alternative to a persistent identifier (*line 3*).

```
1 unFAIR_file:
2   class: File
3   location: https://www.ibi.vu.nl/downloads/PIPENN/PIPENN/BioDL-Datasets/prepared_biolip_win_p_testing.csv
4   format: http://edamontology.org/format_3752 # csv
5   s:additionalType: s:Dataset
6   s:name: "BioDL test set"
7   s:dateModified: "2021-08-02" # an alternative to version
8   s:citation:
9     s:identifier: https://doi.org/10.1093/bioinformatics/btac071
10    s:license: https://spdx.org/licenses/GPL-3.0-or-later.html
11    s:description: "BioDL test set containing only protein-protein interactions."
```

B.2 Annotations requiring extension of CWL Standards

B.2.1 Annotation of a dataset merged from two other datasets. In this example, a dataset is constructed from two other datasets (*lines 4-6*). Here, the instrument is not a tool but a function (*line 7*).

```
1 cwlprov:prov:
2   merge_action:
3     s:additionalType: s:Action
4     s:object:
5       - dataset1
6       - dataset2
7     s:instrument: pd.merge(dataset1, dataset2, on = "ID", how = "inner")
8     s:result: merged_dataset
9     s:description: "Imported both datasets as pandas dataframes, performed inner merge and saved as csv."
10
11 merged_dataset:
12   class: File
13   location: path://path/to/file.csv
```

B.3 Annotations for epitope prediction workflow

Below, we show how we applied our annotation scheme to the input file for our example workflow. We represented the download action which produced the *sabdad_summary_file* (*lines 3-16*). If files were not downloaded during workflow execution, we supplied the URL to the dataset using *s:distribution* (*line 69*). Every file has a citation (*s:citation*), consisting of the DOI of the primary publication. Finally, we supplied EDAM annotations to each *File* and *Directory* input (*lines 29, 44, 58, 72*) and annotated the workflow run (*line 78*).

```
1 cwlprov:prov:
2   sabdad_search:
3     s:additionalType: s:SearchAction
4     s:query: "All structures"
5     s:endTime: 2022-05-27
6     s:object:
7       s:additionalType: s:DataCatalog
8       s:name: "Structural Antibody Database"
9       s:citation:
10        s:identifier: https://doi.org/10.1093/nar/gkab1050
11      s:result: sabdad_summary_file
12      s:description: "Search Action for metadata on antibody-antigen complexes in SAbDab"
13      s:additionalType: edam:operation_0339 # structure database search
14
15 pdb_search_api_query:
16   class: File
17   location: path://path/to/pdb_query.json
```



```

18   format: iana:application/json
19   s:description: "Input query for PDB search API."
20   s:additionalType:
21     - edam:data_3786 # Query script
22
23   sabdab_summary_file:
24     class: File
25     path: path://path/to/sabdab_summary_all_20220527.tsv
26     format: iana:text/tab-separated-values
27     s:description: "Summary file downloaded from SAbDAb database, containing metadata for all structures."
28     s:additionalType:
29       - edam:data_2080 # database search results
30       - s:Dataset
31
32   bioldl_train_dataset:
33     class: File
34     location: https://www.ibi.vu.nl/downloads/PIPENN/PIPENN/BioDL-Datasets/prepared_biolip_win_p_training.csv
35     format: http://edamontology.org/format_3752 # csv
36     s:description: "BioDL training set containing PPI annotations for protein sequences (UniProt IDs)"
37     s:name: "BioDL training dataset"
38     s:dateModified: 2021-08-04
39     s:citation:
40       s:identifier: https://doi.org/10.1093/bioinformatics/btac071
41     s:license: https://spdx.org/licenses/GPL-3.0-or-later.html
42     s:additionalType:
43       - s:Dataset
44       - edam:data_1277 # protein features
45
46   bioldl_test_dataset:
47     class: File
48     location: https://www.ibi.vu.nl/downloads/PIPENN/PIPENN/BioDL-Datasets/prepared_biolip_win_p_testing.csv
49     format: http://edamontology.org/format_3752 # csv
50     s:description: "BioDL test set containing PPI annotations for protein sequences (UniProt IDs)."
51     s:name: "BioDL test dataset"
52     s:dateModified: 2021-08-04
53     s:citation:
54       s:identifier: https://doi.org/10.1093/bioinformatics/btac071
55     s:license: https://spdx.org/licenses/GPL-3.0-or-later.html
56     s:additionalType:
57       - s:Dataset
58       - edam:data_1277 # protein features
59
60   hhblits_db_dir:
61     class: Directory
62     location: path://path/to/uniclust30_2018_08/
63     s:citation:
64       s:identifier: https://doi.org/10.1038/nmeth.1818
65     s:name: "uniclust30_2018_08_hhsuite"
66     s:version: "2018_08"
67     s:description: "Directory containing HHBlits reference database."
68     s:license: https://spdx.org/licenses/CC-BY-SA-4.0
69     s:distribution: https://wwwuser.gwdg.de/~compbiol/uniclust/2018_08/uniclust30_2018_08_hhsuite.tar.gz
70     s:additionalType:
71       - s:Dataset
72       - edam:data_0955 # data index
73
74   hhblits_db_name: "uniclust30_2018"

```

```
75
76 hhblits_n_iterations: 1
77
78 s:description: "Demonstration run of epitope prediction workflow. Some steps are emulated, so the results of
   ↳ the workflow are not yet biologically meaningful."
79
80 $namespaces:
81   iana: "https://www.iana.org/assignments/media-types/"
82   s: "https://schema.org/"
83   edam: "http://edamontology.org/"
84
85 $schemas:
86 - https://schema.org/version/latest/schemaorg-current-https.rdf
87 - https://edamontology.org/EDAM_1.25.owl
```