Recording provenance of workflow runs with RO-Crate

Simone Leo[1*], Michael R. Crusoe[2,3,4], Laura Rodríguez-Navas[5], Raül Sirvent[5], Alexander Kanitz[6,7], Paul De Geest[8], Rudolf Wittner[9,10,11], Luca Pireddu[1], Daniel Garijo[12], José M. Fernández[5], Iacopo Colonnelli[13], Matej Gallo[9], Tazro Ohta[14,15], Hirotaka Suetake[16], Salvador Capella-Gutierrez[5], Renske de Wit[2], Bruno P. Kinoshita[5], Stian Soiland-Reyes[17,18]

1 Center for Advanced Studies, Research, and Development in Sardinia (CRS4), Pula (CA), Italy
2 Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
3 DTL Projects, The Netherlands
4 Forschungszentrum Jülich, Germany
5 Barcelona Supercomputing Center, Barcelona, Spain
6 Biozentrum, University of Basel, Basel, Switzerland
7 Swiss Institute of Bioinformatics, Lausanne, Switzerland
8 VIB Data Core, Gent, Belgium
9 Faculty of Informatics, Masaryk University, Brno, Czech Republic
10 Institute of Computer Science, Masaryk University, Brno, Czech Republic
11 BBMRI-ERIC, Graz, Austria
12 Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
13 Computer Science ~~Dept.~~Department, Università degli Studi di Torino, Torino, Italy
14 Database Center for Life Science, Joint Support-Center for Data Science Research, Research Organization of Information and Systems, Shizuoka, Japan
15 Institute for Advanced Academic Research, Chiba University, Chiba, Japan
16 Sator, ~~Inc.~~Incorporated, Tokyo, Japan
17 Department of Computer Science, The University of Manchester, Manchester, United Kingdom
18 Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands

* simone.leo@crs4.it (SL)

Abstract

Recording the provenance of scientific computation results is key to the support of traceability, reproducibility and quality assessment of data products. Several data models have been explored to address this need, providing representations of workflow plans and their executions as well as means of packaging the resulting information for archiving and sharing. However, existing approaches tend to lack interoperable adoption across workflow management systems. In this work we present Workflow Run RO-Crate, an extension of RO-Crate (Research Object Crate) and Schema.org to capture the provenance of the execution of computational workflows at different levels of granularity and bundle together all their associated ~~products~~ objects (inputs, outputs, code, etc.). The model is supported by a diverse, open community that runs regular meetings, discussing development, maintenance and adoption aspects. Workflow Run RO-Crate is already implemented by several workflow management systems, allowing interoperable comparisons between workflow runs from heterogeneous systems. We describe the model, its alignment to standards such as W3C PROV, and its implementation in six

workflow systems. Finally, we illustrate the application of Workflow Run RO-Crate in two use cases of machine learning in the digital image analysis domain.

1 Introduction

A crucial part of scientific research is recording the provenance of its outputs. The W3C PROV standard defines provenance as "a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing" [1]. Provenance is instrumental to activities such as traceability, reproducibility, accountability, and quality assessment [2]. The constantly growing size and complexity of scientific datasets and the analysis that is required to extract useful information from them has made science increasingly dependent on advanced automated processing techniques in order to get from experimental data to final results [4–6]. Consequently, a large part of the provenance information for scientific outputs consists of descriptions of complex computer-aided data processing steps. This data processing is often expressed as workflows ~~,~~ — i.e., high-level applications that coordinate multiple tools and manage intermediate outputs in order to produce the final results.

In order to homogenise the collection and interchange of provenance records, the W3C consortium proposed ~~the~~ a standard for representing provenance in the Web (PROV [1]), along with the PROV ontology (PROV-O~~standard~~) [7], an OWL [8] representation of PROV~~for provenance in the Web.~~ . PROV-O has been widely extended for workflows (e.g., D-PROV [9], ProvONE [10], OPMW ~~[11]~~ [11] (Open Provenance Model for Workflows), P-PLAN [12]), where provenance information is collected in two main forms: prospective and retrospective [13]. *Prospective provenance* – the execution plan – is essentially the workflow itself: it includes a machine-readable specification with the processing steps to be performed and the data and software dependencies to carry out each computation. *Retrospective provenance* refers to what actually happened during an execution ~~,~~ — i.e. what were the values of the input parameters, which outputs were produced, which tools were executed, how much time did the execution take, whether the execution was successful or not, etc. Retrospective provenance ~~can also~~ may be represented at different levels of abstraction~~depending on available computing resources: for instance, by the workflow execution becoming a single activity which produces results, by specifying the~~ , depending on the information that is available and/or required: a workflow execution may be interpreted i) as a single end-to-end activity, ii) as a set of individual execution of ~~each workflow step, or~~ workflow steps, or iii) by going a step further and indicating how each step is divided into sub-processes when a workflow is deployed in a cluster. ~~Different workflow systems have adopted and extended PROV (~~ Various workflow management systems, such as WINGS [15] (Workflow INstance Generation and Specialization) and VisTrails [18, 19], have adopted PROV and its PROV-O representation ~~) to the workflow domain (WINGS [15, 17], VisTrails [18, 19]), in order to ease the~~ to lift the burden of provenance collection from tool ~~developers to workflow management systems (WMS)~~ users and developers [20, 21].

D-PROV, PROV-ONE, ~~OPMW-PROV, P-Plan~~ OPMW, P-PLAN propose representations of workflow plans and their respective executions, taking into account the features of the workflow systems implementing them (e.g., hierarchical representations, sub-processes, etc.). Other data models~~like~~ , such as *wfprov* and *wfdesc* ~~[22]~~ [22], go a step further by considering not only the link between plans and executions, but also how to package the various artefacts as a Research Object (RO) [23] ~~in order to ease portability while keeping~~ to improve metadata interoperability and document the context of a digital experiment.

However, while these models address some workflow provenance representation issues, they have two main limitations: ~~firstly~~first, the extensions of PROV are not directly interoperable because of differences in ~~granularity~~ their granularities or different assumptions in their workflow representations; ~~secondly~~second, their support from ~~WMS~~ Workflow Management Systems (WMS) is typically one system per model. An early approach to unify and integrate workflow provenance traces across ~~WMS was WEST (~~WMSs was the Workflow Ecosystems through STandards ~~) [17], through the use of WINGS [15]~~ (WEST) [17], which used WINGS to build workflow templates and different converters. In all of these workflow provenance models, the emphasis is on the workflow execution structure as a directed graph, with only partial references for the data items. The REPRODUCE-ME ontology [24] extended PROV and ~~P-Plan~~ P-PLAN to explain the overall scientific process with the experimental context including real life objects (e.g. instruments, specimens) and human activities (e.g. lab protocols, screening), demonstrating provenance of individual Jupyter Notebook cells~~()~~ [25] and highlighting the need for provenance also where there is no workflow management system.

More recently, interoperability ~~have~~ has been partially addressed by Common ~~Worlflow~~ Workflow Language Prov (CWLProv) [26], which represents workflow enactments as ~~ROs~~ research objects serialised according to the Big Data Bag ~~(BDBag)~~ approach [27]. The resulting format is a folder containing several data and metadata files [28], expanding on the ~~RO~~ Research Object Bundle approach of Taverna [29]. CWLProv also extends PROV with a representation of executed processes (activities), their inputs and outputs (entities) and their executors (agents), together with their Common Workflow Language ~~specification~~ (CWL) specification [30] – a standard workflow specification adopted by at least a dozen different workflow systems~~()~~ [31]. Although CWLProv includes prospective provenance as a *plan* within PROV (based on the *wfdesc* model), in practice its implementation does not include tool definitions or file formats~~, as proposed by the wfdesc extension Roterms ().In order~~ . Thus, for CWLProv consumers to reconstruct the full prospective provenance for understanding the workflow, they would also need to inspect the separate workflow definition in the native language of the ~~WMS~~workflow management system. Additionally, the CWLProv RO may include several other metadata files and PROV serialisations conforming to different formats, complicating its generation and consumption.

As for granularity, CWLProv ~~proposed~~ proposes multiple levels of provenance ~~[26, figure 2]~~ [26, Figure 2], from Level 0 (capturing workflow definition) to Level 3 (domain-specific annotations). In practice, the CWL reference implementation *cwltool* [33] and the corresponding CWLProv specification ~~[28] records~~ [28] record provenance details of all task executions together with the intermediate data and any nested workflows (CWLProv level 2)~~, a granularity level that~~ . This level of granularity requires substantial support from the ~~WMS. This approach is~~ workflow management system implementing the CWL specification, resulting appropriate for workflow languages where the execution plan, including its distribution among the various tasks, is well known in advance~~(such as CWL)~~. However, it can be at odds with other systems where the execution is more dynamic, depending on the verification of specific runtime conditions, such as the size and distribution of the data (e.g., COMPSs [34]). This design makes the implementation of CWLProv challenging, ~~as shown by the fact that~~ which the authors suspect may be one of the main causes for the low adoption of CWLProv (at the time of writing the format is supported only by cwltool). Finally, being based on the PROV model, CWLProv is highly focused on the interaction between agents, processes and related entities, while support for contextual metadata (such as workflow authors, licence or creation date) in the Research Object Bundle is limited~~()~~ [35] and stored in a separate manifest file, ~~that~~ which includes the data identifier mapping to filenames. A project that uses serialised ~~ROs~~ Research Objects similar to those used by CWLProv is Whole Tale [36], a web platform with a focus on the narrative around scientific studies and their reproducibility, where the serialised ROs are used to export data and metadata from the platform. In contrast, our work is primarily focused on the ability to capture the provenance of computational workflow execution including its data and executable workflow definitions.

RO-Crate [37] is ~~a recent approach to~~ an approach for packaging research data together with their metadata ~~; it~~ and associated resources. RO-Crate extends Schema.org [38], a popular vocabulary for describing resources on the Web. In its simplest form, an RO-Crate is a directory structure that contains a single JSON-LD [39] metadata file at the top level. The metadata file describes all entities stored in the RO-Crate along with their relationships~~;~~ , and it is both machine-readable and human-readable. RO-Crate is general enough to be able to describe any dataset, but can also be made as specific as needed through the use of extensions called *profiles*. ~~At the same time, the~~ Profiles describe “a set of conventions, types and properties that one minimally can require and expect to be present in that subset of RO-Crates” [105]. The broad set of types and properties from Schema.org, complemented by a few additional terms from other vocabularies, make the RO-Crate model ~~capable of~~ a candidate for expressing a wide range of contextual information that complements and enriches the core information specified by the profile. This information may include, among others, the workflow authors and their affiliations, associated publications, licensing information, related software, etc. This ~~is an approach~~ approach is used by WorkflowHub [40], a ~~workflow system agnostic workflow~~ workflow-system-agnostic workflow registry which specifies a Workflow RO-Crate profile [41] to gather the workflow definition with such metadata in an archived RO-Crate.

In this work, we present **Workflow Run RO-Crate** (WRROC), an extension of RO-Crate for representing computational workflow execution provenance. Our main contributions ~~are the following:~~

include:

- ~~A~~ a collection of RO-Crate profiles to represent and package both the prospective and the retrospective provenance of a computational workflow run in a way that is machine-actionable ~~[42], independent~~ [42], independently of the specific workflow language or execution system, and including support for ~~reexecution.~~ re-execution;
- ~~Implementations of the~~ implementations of this new model in six workflow management systems and in one conversion tool;
- ~~A~~ a mapping of our profiles against the W3C PROV-O Standard using the Simple Knowledge Organisation System (SKOS) ~~[43]~~ [43].

To foster usability, the profiles are characterised by different levels of detail, and the set of mandatory metadata items is kept to a minimum in order to ease the implementation. This flexible approach increases the model’s adaptability to the diverse landscape of ~~WMS~~ WMSs used in practice. The base profile, in particular, is applicable to any kind of computational process, not necessarily described in a formal workflow language. All profiles are supported and sustained by the Workflow Run RO-Crate community, which meets regularly to discuss extensions, issues and new implementations.

The rest of this work is organised as follows: we first describe the Workflow Run RO-Crate profiles in Section 2; we then illustrate implementations in Section 3 and usage examples ~~; this is followed by a discussion and~~ in Section 4; finally, we include a discussion in Section 5 and we conclude the paper with our plans for future work in Section 6.

2 The Workflow Run RO-Crate profiles

RO-Crate profiles are extensions of the base RO-Crate specification that describe how to represent the ~~entities~~ classes and relationships that appear in a specific domain or use case. An RO-Crate conforming to a profile is not just machine-readable, but also machine-actionable, as a digital object whose type is represented by the profile itself [44].

The Workflow Run RO-Crate profiles are the main outcome of the activities of the Workflow Run RO-Crate Community~~()~~ [45], an open working group that includes workflow users and developers, WMS users and developers, and researchers and software engineers interested in workflow execution provenance and Findable, Accessible, Interoperable and Reusable (FAIR) approaches for data and software. ~~In order to develop the~~ One of the first steps in the development of the Workflow-Run RO-Crate profiles ~~, one of the first community efforts~~ was to compile a list of requirements to be addressed by the model from all interested participants, in the form of ~~competency questions ()to be addressed by the model.~~ *competency questions* (CQs) [46]. The process also included reviewing existing state of the art models, such as wfprov [22], ProvONE [10] or OPMW [11]. The result was the definition of 11 CQs capturing requirements which span a broad application scope and consider different levels of provenance granularity. Each requirement was ~~backed up~~ supported by a rationale and linked to a GitHub issue to drive the public discussion forward. When a requirement was addressed, related changes were integrated into the profiles and the relevant issue was closed. ~~Many of~~ All the original issues are now closed, and the profiles have had ~~four~~ five official releases on Zenodo~~.~~ [53, 55, 59]. The target of several of the original CQs evolved during profile development, as the continuous discussion within the community highlighted the main points to be addressed. This continuous process is reflected in the corresponding issues and pull requests in the community's GitHub repository. The final implementation of the CQs in the profiles is validated with SPARQL queries that can be run on RO-Crate metadata samples, also available on the GitHub repository [47].

As requirements were being defined, it became apparent that one single profile would not have been sufficient to cater for all possible usage scenarios. In particular, while some use cases required a detailed description of all computations orchestrated by the workflow, others were only concerned with a "black box" representation of the workflow and its execution as a whole (i.e., whether the ~~execution~~ workflow execution as a whole was successful and which results were obtained). Additionally, some computations involve a data flow across multiple applications that are executed without the aid of a WMS and thus are not formally described in a standard workflow language. These observations led to the development of three profiles: ~~(1) Process Run Crate ()~~

1. *Process Run Crate*, to describe the execution of one or more tools that contribute to a computation; ~~(2) Workflow Run Crate ()~~

2. *Workflow Run Crate*, to describe a computation orchestrated by a predefined workflow; ~~(3) Provenance Run Crate ()~~

3. *Provenance Run Crate*, to describe a workflow computation including the internal details of individual step executions.

In the rest of this section we describe each of ~~the above~~ these profiles in detail. We use the term "class" to refer to a type as defined in RDF(s) and "entity" to refer to an instance of a class. We use italics to denote the ~~types and properties describing entities and their relationships~~properties and classes in each profile: these are defined in the RO-Crate JSON-LD context~~()~~ [49], which extends Schema.org with terms from the Bioschemas [50] ComputationalWorkflow profile~~()~~ [51] and other vocabularies. ~~More~~

~~specifically, from Bioschemas we use the *ComputationalWorkflow* and *FormalParameter* types as well as the *input* and *output* properties. Note that these terms , though~~ Note that terms coming from Bioschemas ~~,~~ are not specific to the life sciences. We also developed a ~~context extension through a dedicated "workflow-run" namespace ()~~ dedicated term set [52] to represent concepts that are not captured by terms in the RO-Crate context. New terms are defined in RDF(s) following Schema.org guidelines (i.e., using *domainIncludes* and *rangeIncludes* to define domains and ranges of properties). In the rest of the text and images, the following prefixes are used to represent the corresponding namespaces:

s:	→	`https://schema.org/`
bioschemas:	→	`https://bioschemas.org/`
bsp:	→	`https://bioschemas.org/properties/`
wfrun:	→	`https://w3id.org/ro/terms/workflow-run#`

2.1 Process Run Crate

The Process Run Crate profile ~~[?] contains specifications on describing~~ [53] contains specifications to describe the execution of one or more software applications that contribute to the same overall computation, but are not necessarily coordinated by a top-level workflow or script ~~. For instance, they could be~~ (e.g. when executed manually by a human~~agent~~, one after the other as intermediate datasets become available~~, as shown in the process run crate ()from [54]).~~).

~~Being~~ The Process Run Crate is the basis for all profiles in the WRROC collection~~, Process Run Crate~~. It specifies how to describe the fundamental ~~entities~~ classes involved in a computational run: i) a software application ~~(~~represented by a ~~*SoftwareApplication*, *SoftwareSourceCode* or *ComputationalWorkflow* entity) and its execution (~~*s:SoftwareApplication*, *s:SoftwareSourceCode* or *bioschemas:ComputationalWorkflow* class; and ii) its execution, represented by a ~~*CreateAction* entity), with the latter~~ *s:CreateAction* class, and linking to the ~~former via the *instrument* property.~~ application via the *s:instrument* property. Other important properties of the ~~*CreateAction* entity are *object*~~*s:CreateAction* class are *s:object*, which links to the action's inputs, and ~~*result*~~*s:result*, which links to its outputs. The time the execution started and ended can be provided, respectively, via the ~~*startTime* and *endTime*~~ *s:startTime* and *s:endTime* properties. The ~~*Person* or *Organization* entity~~ *s:Person* or *s:Organization* class that performed the action is ~~referred to via the *agent*~~ specified via the *s:agent* property. Fig 1 shows the ~~entities~~ classes used in Process Run Crate together with their relationships.

As an example, suppose a user ~~called~~ named John Doe runs the ~~`head` UNIX command~~ UNIX command `head` to extract the first ten lines of an input file named `lines.txt`, storing the result in another file called `selection.txt`. John then runs the `sort` UNIX command on `selection.txt`, storing the sorted output in a new file named `sorted_selection.txt`.

Fig 2 contains a diagram of the two actions and their relationships to the other ~~entities involved~~ involved entities. Note how the actions are connected by the fact that the output of "Run Head" is also the input of "Run Sort": they form an "implicit workflow", whose steps have been executed manually rather than by a software tool.

Process Run Crate extends the RO-Crate guidelines on representing software used to create files with additional requirements and conventions. This arrangement is typical of the RO-Crate approach, where the base specification provides general recommendations to allow for high flexibility, while profiles – being more concerned with the representation of specific domains and machine actionability – provide more detailed and structured definitions. Nevertheless, in order to be broadly applicable, profiles also need to avoid the specification of too many strict requirements, trying to strike a good

Fig 1. UML class diagram for Process Run Crate. The central ~~entity~~ class is the ~~*CreateAction*~~*s:CreateAction*, which represents the execution of an application. It ~~relates with~~ links to the application itself via ~~*instrument*~~*s:instrument*, ~~with~~ to the entity that executed it via ~~*agent*~~ *s:agent*, and ~~with~~ to its inputs and outputs via ~~*object*~~ *s:object* and ~~*result*~~*s:result*, respectively. ~~*File* is an RO-Crate alias for Schema~~In this and following figures, classes and properties are shown with prefixes to indicate their origin. ~~org's *MediaObject*.~~ Some inputs (and, less commonly, outputs) ~~, however,~~ are not stored as files or directories, but passed to the application (e.g., via a command line interface) as values of various types (e.g., a number or string). In this case, the profile recommends a representation via ~~*PropertyValue*~~*s:PropertyValue*. For simplicity, we left out the rest of the RO-Crate structure (e.g. the root ~~*Dataset*~~*s:Dataset*), and attributes (e.g. *s:startTime*, *s:endTime*, *s:description*, *s:actionStatus*). In this UML class notation, diamond ◊ arrows indicate aggregation and regular arrows indicate references, ∗ indicates ~~multiple instances~~zero or more occurrences, 1 means single ~~instance~~occurrence.

Fig 2. Diagram of a simple workflow where the `head` and `sort` programs were run manually by a user. The executions of the individual software programs are connected by the fact that the file output by `head` was used as input for `sort`, documenting the computational flow in an implicit way. Such executions can be represented with Process Run Crate.

trade-off between flexibility and actionability. ~~One of the implications of this approach is that consumers need to code defensively, avoiding unwarranted assumptions – e.g. by verifying that a value exists for an optional property before trying to retrieve it and use it.~~

2.2 Workflow Run Crate

The Workflow Run Crate profile ~~[?]~~ [55] combines the Process Run Crate and WorkflowHub's Workflow RO-Crate [41] profiles to describe the execution of ~~"proper"~~ computational workflows managed by a WMS. Such workflows are typically written in a ~~special-purpose~~ domain-specific language, such as CWL or Snakemake [56], and run by one or more WMS (e.g., StreamFlow [57], Galaxy [58]). Fig 3 illustrates the classes used in this profile together with their relationships. As in Process Run Crate, the execution is described by a ~~*CreateAction*~~ *s:CreateAction* that links to the application via ~~*instrument*~~*s:instrument*, but in this case the application must be a workflow, as prescribed by Workflow RO-Crate. More specifically, Workflow RO-Crate states that the RO-Crate must contain a main workflow typed as *File* ~~, *SoftwareSourceCode* and *ComputationalWorkflow*~~(an RO-Crate mapping to *s:MediaObject*), *s:SoftwareSourceCode* and *bioschemas:ComputationalWorkflow*. The execution of the individual workflow steps, instead, is not represented: that is left to the more detailed Provenance Run Crate profile (described in the next section).

The Workflow Run ~~RO-Crate~~ Crate profile also contains recommendations on how to represent the workflow's input and output parameters, based on the ~~aforementioned Bioschemas [50]~~ Bioschemas ComputationalWorkflow profile. All these elements are represented via the ~~*FormalParameter* entity~~ *bioschemas:FormalParameter* class and are referenced from the main workflow via the ~~*input* and *output*~~ *bsp:input* and *bsp:output* properties. While the ~~entities referenced from *object* and *result* in the *CreateAction*~~ classes referenced from *s:object* and *s:result* in the *s:CreateAction* represent data entities and argument values that were actually used in the workflow execution, the ones referenced from ~~*input* and *output*~~ *bsp:input* and *bsp:output* correspond to formal parameters, which acquire a value when the workflow is run (see Fig~~.~~ 3). In the profile, the relationship between an actual value and the corresponding formal parameter is expressed through the ~~*exampleOfWork* property—the downloadable file is a realisation of the formal parameter definition~~*s:exampleOfWork* property. For instance, in the following JSON-LD snippet a formal parameter (`#annotations`) is illustrated together with a corresponding `final-annotations.tsv` file:

```
{
    "@id": "#annotations",
    "@type": "FormalParameter",
    "additionalType": "File",
    "encodingFormat": "text/tab-separated-values",
    "valueRequired": "True",
    "name": "annotations"
},
{
    "@id": "final-annotations.tsv",
    "@type": "File",
    "contentSize": "14784",
    "exampleOfWork": {"@id": "#annotations"}
}
```

~~The derivation of Workflow Run Crate from Workflow RO-Crate makes RO-Crates that conform to this profile compatible with the WorkflowHub workflow registry by also conforming to its Workflow RO-Crate profile. Thus, users of a WMS that implements this profile (or Provenance Run Crate, which inherits it) are able to register their workflows in WorkflowHub—together with an execution trace—by simply running them and uploading the resulting RO-Crates. Additionally, the inheritance mechanism allows to reuse the specifications already developed for Workflow RO-Crate, which form part of the guidelines on representing the prospective provenance.~~

~~Fig 3 shows the entities used in Workflow Run Crate together with their relationships.~~

Fig 3. UML class diagram for Workflow Run Crate. The main differences with Process Run Crate are the representation of formal parameters and the fact that the ~~application~~ workflow is expected to be an entity with types *s:MediaObject* (*File* in RO-Crate JSON-LD), ~~*SoftwareSourceCode*~~ *s:SoftwareSourceCode* and ~~*ComputationalWorkflow*~~*bioschemas:ComputationalWorkflow*. Effectively, the ~~entity~~ workflow belongs to all three types, and its properties are the union of the properties of the individual types. In this profile, the execution history (retrospective provenance) is augmented by a (prospective) workflow definition, giving a high-level overview of the workflow and its input and output parameter definitions (*bioschemas:FormalParameter*). The inner structure of the workflow is not represented in this profile. In the provenance part, individual files (*s:MediaObject*) or arguments (*s:PropertyValue*) are then connected to the parameters they realise. Most workflow systems can consume and produce multiple files, and this mechanism helps to declare each file's role in the workflow execution. The filled diamond ♦ indicates composition, empty diamond ◊ aggregation, and other arrows relations.

2.3 Provenance Run Crate

The Provenance Run Crate profile~~[?]~~ [59] extends Workflow Run Crate by adding new concepts to describe the internal details of a workflow run, including individual tool executions, intermediate outputs and related parameters. Individual tool executions are represented by additional ~~*CreateAction*~~ *s:CreateAction* instances that refer to the tool itself via ~~*instrument*~~ *s:instrument* – analogously to its use in Process Run Crate. The workflow is required to refer to the tools it orchestrates through the ~~*hasPart*~~ *s:hasPart* property, as suggested in the Bioschemas ComputationalWorkflow profile, though in the latter it is only a recommendation.

To represent the logical steps defined by the workflow, this profile uses ~~*HowToStep*~~ ~~i.e., “~~*s:HowToStep* – i.e., “A step in the instructions for how to achieve a result~~”~~ ~~()~~” [60]. Steps point to the corresponding tools via the ~~*workExample*~~ *s:workExample* property and are referenced from the workflow via the ~~*step*~~ *s:step* property; the execution of a step is represented by a ~~*ControlAction*~~ *s:ControlAction* pointing to the ~~*HowToStep* via *instrument*~~ *s:HowToStep* via *s:instrument* and to the ~~*CreateAction* instance(s)~~ *s:CreateAction* entities that represent the corresponding tool execution(s) via ~~*objects*~~*s:object*. Note that a step execution does not coincide with a tool execution: an example where this distinction is apparent is when a step maps to multiple executions of the same tool over a list of inputs (e.g. the “scattering” feature in CWL).

An RO-Crate following this profile can also represent the execution of the WMS itself (e.g., cwltool) via ~~*OrganizeAction*~~*s:OrganizeAction*, pointing to a representation of the WMS via ~~*instruments*~~*s:instrument*, to the steps via ~~*object*~~ *s:object* and to the workflow run via ~~*result*. The *object*~~ *s:result*. The *s:object* attribute of the ~~*OrganizeAction*~~ *s:OrganizeAction* can additionally point to a configuration file containing a description of the settings that affected the behaviour of the WMS during the execution.

Fig 4 ~~shows the various entities~~ illustrates the various classes involved in the representation of a workflow run via Provenance Run Crate together with their relationships.

Fig 4. UML class diagram for Provenance Run Crate. In addition to the workflow run, this profile represents the execution of individual steps and their related tools. The prospective side (the execution plan) is shown by the workflow listing a series of *s:HowToStep*s, each linking to the *s:SoftwareApplication* that is to be executed. The *bsp:input* and *bsp:output* parameters for each tool are described in a similar way to the overall workflow parameter in Fig 3. The retrospective provenance side of this profile includes each tool execution as an additional *s:CreateAction* with similar mapping to the realised parameters as *s:MediaObject* or *s:PropertyValue*, allowing intermediate values to be included in the RO-Crate even if they are not workflow outputs. The workflow execution is described the same as in the Workflow Run Crate profile with an overall *s:CreateAction* (the workflow outputs will typically also appear as outputs from inner tool executions). An additional *s:OrganizeAction* represents the workflow engine execution, which orchestrated the steps from the workflow plan through corresponding *s:ControlAction*s that spawned the tool’s execution (*s:CreateAction*). It is possible that a single workflow step had multiple such executions (e.g. array iterations). Not shown in figure: *s:actionStatus* and *s:error* to indicate step/workflow execution status. The filled diamond ♦ indicates composition, empty diamond ◊ aggregation, and other arrows relations.

~~This profile also includes specifications on~~ Additionally, this profile specifies how to describe connections between parameters~~. Parameter connections~~, through *parameter connections* – a fundamental feature of computational workflows~~– describe~~. Specifically, parameter connections describe: (i) how tools ~~take~~ consume as input the intermediate outputs generated by other tools; and (ii) how workflow-level parameters are mapped to tool-level parameters. ~~For instance~~As an example, consider again the workflow depicted in Fig. 2, and suppose it is implemented in a workflow language such as CWL~~. The~~: the workflow-level input (a text file) is ~~connected~~ linked through a parameter connection to the input of the ~~"head"~~ `head` tool wrapper, and ~~the output of the latter is connected~~ then a second parameter connection links this tool's output to the input of the ~~"sort"~~ `sort` tool wrapper.

A representation of parameter connections is particularly useful for traceability, since it ~~allows~~ provides the means to document the inputs and tools on which workflow outputs depend. Since the current RO-Crate context has no suitable terms for the description of such relationships, we added appropriate ones to the aforementioned ~~"workflow-run" context extension (the namespace): a *ParameterConnection* type with *sourceParameter* and *targetParameter*~~ dedicated term set [52]: a *wfrun:ParameterConnection* type with *wfrun:sourceParameter* and *wfrun:targetParameter* attributes that respectively map to the source and target formal parameters, and a ~~*connection*~~ *wfrun:connection* property to link from the relevant step or workflow to the ~~*ParameterConnection*~~ *wfrun:ParameterConnection* instances.

~~This profile~~ In our set of profiles, Provenance Run Crate is the most detailed ~~of the three,~~ one and offers the highest level of granularity~~. Fig. 5shows the relationship between the specifications of the profiles as a Venn diagram.~~ ; its specification is a superset of Workflow Run RO-Crate, which in turn is a superset of Process Run Crate. This relationship between the three profiles is illustrated in Fig 5, as a Venn diagram. Theoretically, all computational provenance information could be represented through the Provenance Run Crate profile alone (possibly relaxing some requirements), since it inherits from the other ones. In practice, though, this choice would require the use of the most complex model even for simple use cases. Having three separate profiles provides a way to represent information at different levels of granularity, while keeping all RO-Crates generated with them interoperable. This approach gives a straightforward path to supporting the representation of computational provenance in simpler use cases such as with simple command executions, i.e. the Process Run Crate. Additionally, the approach lowers the accessibility barrier for implementation in WMSs, as developers may choose to initially implement only the more basic support in their WMS, with reduced effort and complexity, and gradually scale to more detailed representations. This encourages the adoption of WRROC across the diverse landscape of use cases and WMSs.

Fig 5. Venn diagram of the specifications for the various RO-Crate profiles. Process Run Crate specifies how to describe the fundamental classes involved in a computational run, and thus is the basis for all profiles in the WRROC collection. Workflow Run Crate inherits the specifications of both Process Run Crate and Workflow RO-Crate. Provenance Run Crate, in turn, inherits the specifications of Workflow Run Crate (and in a sense includes multiple Process Runs for each step execution, but within a single Crate).

2.4 Profile formats

The WRROC profiles are available both in human-readable (HTML) and in machine-readable format (RO-Crate). The human-readable profiles are at:

- `https://w3id.org/ro/wfrun/process/0.5`
- `https://w3id.org/ro/wfrun/workflow/0.5`

- `https://w3id.org/ro/wfrun/provenance/0.5`

And the corresponding machine-readable ones at:

- `https://doi.org/10.5281/zenodo.12158562`
- `https://doi.org/10.5281/zenodo.12159311`
- `https://doi.org/10.5281/zenodo.12160782`

The RO-Crate metadata files for the machine readable profiles can be retrieved using the same URLs as the human-readable ones, but with JSON-LD content negotiation: this is done by setting `"Accept:application/ld+json"` in the HTTP header.

The new terms we defined to represent concepts that could not be expressed with existing Schema.org ones are at:

- `https://w3id.org/ro/terms/workflow-run`

These terms are available in multiple formats with content negotiation, as explained at the above link.

3 Implementations

Support for the Workflow Run RO-Crate profiles presented in this work has been implemented in a number of systems, showing support from the community and demonstrating their usability in practice. We describe seven of these implementations (one in a conversion tool and six in WMS) in the following sections. Table 1 provides an overview of the implementations, along with the respective profile implemented, and links to the implementation itself and to an example RO-Crate. These tools have been developed in parallel by different teams, and independently from each other. RO-Crate has a strong ecosystem of tools [37], and ~~these~~ the WRROC implementations have either re-used these or added their own approach to the standards.

3.1 Runcrate

Runcrate~~()~~ [61] is a Workflow Run RO-Crate toolkit which also serves as a reference implementation of the proposed profiles. It consists of a Python package with a command line interface, providing a straightforward path to integration in Python software and other workflows. The runcrate toolkit includes functionality to convert CWLProv ROs to RO-Crates conforming to the Provenance Run Crate profile (~~*runcrate convert*~~`runcrate convert`), effectively providing an indirect implementation of the format for cwltool. Indeed, the CWLProv model provided a basis for the Provenance Run Crate profile, and the implementation of a conversion tool in runcrate at times drove the improvement and extension of the profile as new requirements or gaps in the old designs emerged. Runcrate converts both the retrospective provenance part of the CWLProv RO (the RDF graph of the workflow's execution) and the prospective provenance part (the CWL files, including the workflow itself). Both parts are thus converted into a single, ~~workflow language-agnostic~~ workflow-language-agnostic metadata resource.

Another functionality offered by the runcrate package is ~~*runcrate report*~~ `runcrate report`, which reports on the various executions described in an input RO-Crate, listing their starting and ending times, the values of the various parameters, etc. Runcrate report demonstrates how the provenance profiles presented in this work enable comparison of runs interoperably across different workflow languages or different implementations of the same language. This functionality has also been used as a lightweight validator for the various implementations.

~~We also added a *run*~~ Runcrate also includes a `run` subcommand to re-execute the computation described by an input Workflow Run Crate or Provenance Run Crate where CWL ~~was~~ is used as a workflow language. It works by mapping the RO-Crate description of input parameters and their values (the workflow's ~~*input*~~ *bsp:input* and the action's ~~*object*~~*s:object*) to the format expected by CWL, which is then used to relaunch the workflow on the input data. This functionality shows the machine-actionability of the profiles to support reproducibility, and was used to successfully re-execute the digital pathology annotation workflow described in ~~section 4.1.~~

Section 4.1. Of course, achieving a full re-execution in the general case may not always be possible: reproducibility is supported by the profiles, but also benefits from ~~the~~ specific characteristics of the workflow language (which should provide a clear formalism to map input items to their corresponding parameter slots) and ~~from cooperation on the part of the~~ of the specific workflow's ~~author, who can help considerably by containerizing the~~ implementation, which can be made considerably easier to reproduce by containerising the computational environment required by each step ~~and providing the relevant annotations~~ (if allowed by the workflow language).

3.2 Galaxy

The Galaxy project [58] provides a WMS with data management functionalities as a multi-user platform, aiming to make computational biology more accessible to research scientists that do not have computer programming or systems administration experience. Galaxy's most prominent features include: a collection of 7500+ integrated tools~~()~~ [62]; a web interface that allows the ~~execution and definition~~ definition and execution of workflows using the integrated tools; a network of dedicated (public) Galaxy instances.

The export of workflow execution provenance data as Workflow Run Crates ~~has been added in Galaxy 's 23.0 release. This feature provides~~ was added to Galaxy in version 23 [100] providing a more interoperable alternative to the basic export of Galaxy workflow *invocations*. A WRROC export from Galaxy includes: the workflow definition; a set of serialisations of the invocation-related metadata in Galaxy native, ~~json-formatted~~ JSON-formatted files; and the input and output data files. This result is achieved by~~extracting provenance~~ : i) extracting provenance data from Galaxy entities related to the workflow run, along with ~~associated metadata,~~ their associated metadata; ii) converting them to RO-Crate metadata using the ro-crate-py library [63]; ~~by~~ iii) describing all files contained in the basic invocation export within the ~~RO-crate~~ RO-Crate metadata; and ~~finally by~~ iv) making the Workflow Run Crate available for export to the user through Galaxy's web interface and API [64]. We extract the prospective provenance contained in Galaxy's YAML-based gxformat2 ~~()~~ [65] workflow definition, which includes details of the analysis pipeline such as the graph of the tools that need to be executed ~~,~~ and metadata about the data types required. The retrospective provenance – i.e., the details of the executed workflow, such as the inputs, outputs, and parameter values used – is extracted from Galaxy's data model~~(),~~ , which is not directly accessible to users in the context of a public Galaxy server. All of this provenance information is then mapped to RO-Crate metadata, including some Galaxy-specific data entities such as dataset collections. An exemplary ~~exported Galaxy~~ Workflow Run Crate exported from Galaxy, through its *Workflow Invocations* list, is available on Zenodo [66].

In practice, a user would take the following steps to obtain a Workflow Run Crate from a Galaxy instance: ~~(1)~~ i) create or download a Galaxy workflow definition (e.g.: from WorkflowHub) and import it in a Galaxy instance, or create a workflow through the Galaxy GUI directly; ~~(2)~~ ii) execute the workflow, providing the required inputs;

~~(3)~~ iii) after the workflow has run successfully, the corresponding RO-Crate will be available for export from the Workflow Invocations list.

3.3 COMPSs

COMPSs [34] is a task-based programming model that allows users to transform a sequential application into a parallel one by simply annotating some of its methods, thus ~~making it efficient to exploit the resourcesavailable (either distributed or in a cluster)~~facilitating scaling applications to increasing amounts of computing resources. When a COMPSs application is executed, a corresponding workflow describing the application's tasks and their data dependencies is dynamically generated and used by the COMPSs runtime to orchestrate the execution of the application in the infrastructure. As a WMS, COMPSs stands out for its many advanced features that enable applications to achieve fine-grained high efficiency in HPC systems, such as the ability to exploit underlying parallelisation frameworks (~~i.e. MPI, OpenMP~~e.g. MPI [67], OpenMP [68]), compilers (e.g. NUMBA [69]), failure management, task grouping, and more.

~~Provenance~~ Also, provenance recording for COMPSs workflows has been explored in previous work [70], where the Workflow RO-Crate profile was ~~adopted in the implementation of a very lightweight approach to document provenance while avoiding the introduction of~~ used to capture structured descriptive metadata about the executed workflow, without introducing any significant run time performance overheads. ~~However, because of the~~

In this work, COMPSs has been further improved by implementing the generation of provenance information conformant to the Workflow Run Crate profile, thus also capturing details about the actual execution of the workflow. The dynamic nature of COMPSs workflows ~~, the Workflow Run Crate profile is better suited to represent them, since workflows are~~ poses some challenges to capturing provenance, which were met thanks to the instruments provided by the WRROC model. For instance, a COMPSs workflow is created when the application is executed and, thus, a prior static workflow definition does not exist before that moment. Due to this ~~limitation~~design, the workflow entity in the metadata file references the entry point application run by COMPSs ~~, and~~ – instead of, for instance, a dedicated workflow definition file as one might find with other WMSs. Also, formal parameters are not ~~listed~~ included in the prospective provenance (note that ~~listing~~ specifying them is not required by the profile) because inputs and outputs (both for each task and the whole workflow) are determined at runtime. ~~COMPSs is able to export provenance data with a post-processing operation that can be triggered at any moment after the application has finished. The~~ However, the RO-Crate generation ~~post-process uses~~ by COMPSs leverages the information recorded by the runtime to ~~detect and~~ automatically add metadata of ~~any~~ all input or output data assets used or produced by the workflow.

~~Implementing~~ Because of the supercomputing environments where COMPSs is used, the integration of Workflow Run Crate support ~~in COMPSs required~~ required paying particular attention to the generation of a unique ~~id for the *CreateAction*~~ ID for the *s:CreateAction* representing the workflow run~~, combining~~ . Our implementation uses UUIDs for distributed environments, while it adds a combination of hostname and queuing system job ~~id~~ ID for supercomputer executions~~(as extra information added), and just using generated UUIDs for distributed environments, to add~~ , to provide as much information as ~~available~~ possible from the run while ~~ensuring the id is unique~~preserving ID uniqueness. In the ~~*CreateAction, the description*~~ *s:CreateAction*, the *s:description* term includes system information, as well as relevant environment variables that provide details on the execution environment (e.g., node list, CPUs per node). Finally, the ~~*subjectOf*~~ *s:subjectOf* property of the ~~*CreateAction*~~ *s:CreateAction* references the system's monitoring tool (when available), where authorised users can see detailed profiling of the corresponding job execution, as provided by the MareNostrum IV supercomputer~~()~~ [71].

To showcase the COMPSs adoption of the Workflow Run Crate profile, we provide as an example the execution of the BackTrackBB [72] application in the MareNostrum IV supercomputer. BackTrackBB targets the detection and location of seismic sources using the statistical coherence of the wave field recorded by seismic networks and antennas. The resulting RO-Crate [73] captures the provenance of the execution results and complies with the Workflow Run Crate profile~~, and~~. It includes the application source files, a diagram of the workflow's graph, application profiling and input and output files.

The implementation of provenance recording ~~following~~ using Workflow Run Crate has been fully integrated in the COMPSs runtime ~~,~~and is available ~~since~~ as of release 3.2~~[74] ()~~ [74].

3.4 StreamFlow

The StreamFlow framework [57] ~~()~~ is a container-native WMS ~~based on the CWL standard~~for the execution of workflows defined in CWL. It has been designed around two primary principles: first, it allows the execution of tasks in multi-container environments, supporting the concurrent execution of communicating tasks in a multi-agent ecosystem; second, it relaxes the requirement of a single shared data space, allowing for hybrid workflow executions on top of multi-cloud, hybrid cloud/HPC, and federated infrastructures. StreamFlow orchestrates hybrid workflows by combining a *workflow description* (e.g., a CWL workflow description and a set of input values) with one or more *deployment descriptions* – i.e. representations of the execution environments in terms of infrastructure-as-code (e.g., Docker Compose files [75], HPC batch scripts, and Helm charts [76]). A `streamflow.yml` file – the entry point of each StreamFlow execution – binds each workflow step with the set of most suitable execution environments. At execution time, StreamFlow automatically takes care of all the secondary aspects, like scheduling, checkpointing, fault tolerance, and data movements.

StreamFlow ~~stores~~ collects prospective and retrospective provenance data in a ~~proprietary format into a persistent~~ custom format and persists it into a pluggable database (using sqlite3 as the default choice). After a CWL workflow execution completes, users can generate an RO-Crate through the `streamflow prov`~~<workflow_name>~~ command, which extracts the provenance data stored in the database for one or more workflow executions and converts it to an RO-Crate archive that is fully compliant with the Provenance Run Crate Profile, including the details of each task run by the WMS. Support for the format has been integrated into the main development branch and will be included in release 0.2.0 [77].

From the StreamFlow point of view, the main limitation in the actual version of the Provenance Run Crate standard is the lack of support for distributed provenance ~~,~~ – i.e., a standard way to describe the set of locations involved in a workflow execution and their topology. As a temporary solution, the StreamFlow configuration and a description of the hybrid execution environment are preserved by directly including the `streamflow.yml` file into the generated archive. However, this product-specific solution prevents a wider adoption from other WMS and forces agnostic frameworks (e.g., WorkflowHub) to provide ad-hoc plugins to interpret the StreamFlow format. Since the support for hybrid and cross-facility workflows is gaining traction in the WMS ecosystem, we envision support for distributed provenance as a feature for future versions of Workflow Run RO-Crate.

3.5 WfExS-backend

WfExS-backend~~()~~ [78] is a FAIR workflow execution orchestrator that aims to address some of the difficulties found in analysis reproducibility and analysis of sensitive data in a secure manner. WfExS-backend requires that the software used by workflow steps is available in publicly ~~available~~ accessible software containers for reproducibility. Actual workflow execution is delegated to one of the supported workflow engines ~~which matches with the workflow, right now either Nextflow~~ – currently either Nextflow [79] or cwltool. The orchestrator prepares and stages all the elements needed to run the workflow – i.e. all the files of the workflow itself, the specific version of the workflow engine, the required software containers and the inputs. All these elements are ~~referred~~ referenced through resolvable identifiers, ideally public, permanent ones. ~~Due to this~~ Thanks to this approach, the orchestrator can consume workflows ~~which are originally available in different kinds of locations, like~~ from various types of sources, such as git repositories, Software Heritage, or even RO-Crates from WorkflowHub.

WfExS-backend development milestones ~~aim~~ have aimed to reach FAIR workflow execution through the generation and consumption of RO-Crates following the ~~latest~~ Workflow Run Crate ~~profiles, which have~~ profile, which has proven to be a mechanism suitable to semantically describe digital objects in a way that simplifies embedding ~~key details involved in~~ details crucial to analysis reproducibility and replicability.

~~The orchestrator~~ When the orchestrator prepares a workflow for execution it records details relevant to the prospective provenance~~when a workflow is prepared for execution~~, such as the public URLs used to fetch input data and workflows, content digestion fingerprints (typically sha256 checksums) and metadata derived from workflow files, container images and input files. Most of this ~~metadata is represented~~ captured metadata is later included in the generated RO-Crates. WfExS-backend has explicit commands to generate and publish both prospective and retrospective provenance RO-Crates based on a given existing staged execution scenario. These RO-Crates can selectively include copies of used elements as payloads. Workflows can be executed more than once in the same staged directory, with all the executions sharing the same inputs. ~~Thus~~In this case, run details from all the executions are represented in the retrospective provenance RO-Crate. Support for ~~Workflow Run RO-Crate is available since~~ the consumption of Workflow Run RO-Crates to reproduce the operations they document is available as of WfExS-backend version ~~0.10.1 [102]. Future developments~~ 1.0.0a0 [78]. We have created examples of Workflow Run Crates generated by WfExS-backend to capture provenance information from the execution of a Nextflow workflow [80] and a CWL workflow [33]; these crates are both available on Zenodo [81,82]. Future developments to WfExS-backend will also add support for embedding in the RO-Crates the URLs of output results that have been deposited into a suitable repository (like Zenodo DOIs, for instance)~~as well as consuming previously produced RO-Crates~~.

~~An example of Workflow Run Crate generated by WfExS-backend from a Nextflow workflow run [80] is available from Zenodo [?].~~

3.6 Sapporo

Sapporo [83] is an implementation of the Workflow Execution Service (WES) API specification~~().~~ [114]. WES is a standard proposed by the Global Alliance for Genomics and Health (GA4GH) for cloud-based data analysis platforms that receive requests to execute workflows. Sapporo supports the execution of several workflow engines, including cwltool [33], Toil [84], and StreamFlow [57]. Sapporo includes features specifically tailored to bioinformatics applications, including the calculation of feature statistics from specific types of outputs generated by workflow runs. For example, the system calculates the mapping rate of DNA sequence alignments from BAM format files. To describe the feature values, Sapporo uses the Workflow Run Crate profile extended with additional terms to represent these biological features~~()~~ [85].

Further, the Tonkaz companion command line software has integrated functionality to compare Run Crates generated by Sapporo to measure the reproducibility of the workflow outputs [86]. Developers can use this unique feature to build a CI/CD platform for their workflows to ensure that changes to the product do not produce an unexpected result. Workflow users can also use this feature to verify the results from the same workflow deployed in different environments.

While Sapporo supports Workflow Run Crate, since WES is a WMS wrapper, it does not parse the provided workflow definition files. Instead, it embeds the information in the files passed by the WES request to record the provenance of execution rather than using the actual workflow parameters meant for the wrapped WMS. Therefore, the current implementation of Sapporo does not capture the connections between the inputs/outputs depicted in the workflow and the actual files used/generated during the run. ~~Thus, the~~ The profile generated by Sapporo has fields representing input and output files, but they are not linked to formal parameters.

Sapporo supports export to Workflow Run Crate ~~since~~ as of release 1.5.1 [87]. An example of a Workflow Run RO-Crate generated by Sapporo is available on Zenodo [88].

3.7 Autosubmit

Autosubmit [89] is an open source, lightweight workflow manager and meta-scheduler ~~created in 2011 for use in climate research to configure and run scientific experiments~~ tailored to configuring and running scientific experiments in climate research. It supports scheduling jobs via SSH to Slurm [90], PBS [91] and other remote batch servers used in HPC.

~~The~~ Autosubmit's “archive” feature ~~was added in Autosubmit 3.1.0, released in 2015 (). This feature~~ archives the experiment directory and all its contents into a ZIP file, which can be used later to access the provenance data or to execute the Autosubmit experiment again. Even though the data in the ZIP file includes prospective provenance and retrospective provenance, it ~~contains no structure, and users have~~ is not structured, and a simple examination yields no way to ~~tell which is which from just looking at the ZIP file and its contents~~distinguish the provenance types.

Recent releases of Autosubmit 4 ~~include an~~ have added features to increase user flexibility. An updated YAML configuration management system has been implemented that allows users to combine multiple YAML files into a single unified configuration file. ~~While this gave users flexibility, it also increased the complexity to track the configuration changes and to relate these to the provenance data. Another feature added in Autosubmit 4 is the~~ Also, the option to use only the experiment manager features of Autosubmit has been added, delegating the workflow execution to a different backend workflow engine ~~,~~ — like ecFlow [92], Cylc [93], or a CWL runner. While these features provide some much appreciated flexibility, they have increased the complexity involved in reliably tracking the experiment configuration and other metadata for provenance documentation purposes.

In order to give users a more structured way to archive provenance, which includes the complete experiment configuration~~and~~ , the parameters used to generate ~~the unified experiment configuration, and also to allow interoperability~~ it, and is also interoperable between workflow managers, the archive feature ~~received a new flag~~ was enhanced with a new option in Autosubmit 4.0.100 ~~[94] to generate~~ [94] to enable the generation of provenance data in Workflow Run RO-Crates.

The prospective provenance data for the crate is extracted from the Autosubmit experiment configuration. This data includes the multiple YAML files, ~~and~~ the unified YAML configuration, as well as the parameters used to preprocess each file – preprocessing replaces placeholders in script templates with values from the experiment configuration. The retrospective provenance data is included with the RO-Crate archive and includes logs and other traces produced by the experiment workflow. Both prospective and retrospective provenance data are included in the final RO-Crate~~JSON-LD metadata file. Autosubmit uses~~ , which is compliant with the Workflow Run ~~RO-Crate profile.~~

~~As one of the most recent implementations, much of the code added in Autosubmit 4 for RO-Crates was adapted from existing implementations like COMPSs and StreamFlow. ro-crate-py [63] was used for~~ Crate profile. At a practical level, the ~~heavy lifting work of creating~~ implementation was able to leverage the `ro-crate-py` library for many of the details pertaining to the creation of the RO-Crate archive in Python, and adding information for the JSON-LD metadata.

~~The~~ One of the main challenges for ~~adopting RO-Crate in Autosubmit were~~ implementing WRROC support in Autosubmit was incorporating Autosubmit's ~~“Project” feature, and the lack of validation tools and of documentation and examples on how to use the standard with *coarse-grained* workflow management systems (as described in [95]) that do not track inputs and outputs, which is the case of Autosubmit – as well as the Cylc and ecFlow workflow engines.~~

Project feature. A Project in Autosubmit is an abstract concept that ~~has a type and a location, and~~ references a code repository and is used to ~~separate~~ define experiment configuration and ~~template scripts~~ contains template scripts defining workflow tasks and other auxiliary files~~The type can be Git, Subversion, or Local. For each type the location represents the URL of a code repository, or a directory on a workstation or HPC file system used to copy the Project and its template scripts (written in Shell, R, or Python) and any other files (input data for a model, extra configuration files, binaries, etc.). The workflows in Autosubmit have tasks with dependencies to other tasks, and each of these tasks execute one of these template scripts~~. The project has a type that defines the *type* of the repository (e.g., git) and a *location* that is the URL to retrieve it. The RO-Crate file generated by Autosubmit includes ~~only~~ the project type and location, ~~and not~~ but it does not include the complete Project and so it is lacking configuration details and scripts. Therefore, users ~~have the provenance~~ receive provenance data of the Project, but only those with the ~~correct permissions~~ appropriate privileges can access its constituent resources (many applications run with Autosubmit can not be publicly shared without consent).

~~Validation tools for RO-Crate archives are still under development, and while there is a community-based review process to help and guide new implementations, a tool that others can use as code is written will contribute to a more agile development.~~

~~After working~~ After consulting with the RO-Crate community ~~on these issues~~regarding the specific Autosubmit requirements, the Autosubmit team adopted a mixed approach where Autosubmit initialises the JSON-LD metadata from its configuration and local trace files, and the user is responsible for providing a partial JSON-LD metadata object in the Autosubmit YAML configuration. ~~A pull request was created to ro-crate-py to~~ `ro-crate-py` was extended to allow the RO-Crate JSON-LD metadata to be patched by these partial JSON-LD metadata objects. This way, users are able to provide the ~~missing information~~ information that is missing from the Autosubmit configuration model, ~~like~~ but is required by WRROC – e.g., licence, authors, inputs, outputs, formal parameters, ~~and more. And by modifying~~ etc.

Future implementations of WRROC support should be facilitated by the new functionality added to ro-crate-py ~~, future implementers of~~ to support the user-mediated metadata integration approach. On the other hand, the integration of WRROC support would have been facilitated by an automated validation tool for RO-Crate ~~that have a similar workflow configuration as Autosubmit should be able to re-use it, while also using COMPSs, StreamFlow, Autosubmit , and other implementations as reference.~~ archives, and by documentation and examples on how to use the profiles with *coarse-grained* workflow management systems (as defined in [95]) that do not track inputs and outputs, which is the case of Autosubmit – as well as the Cylc and ecFlow workflow engines. The feedback generated by this use case was welcomed by the WRROC community and work to address these issues is either planned on under way at the time of writing.

~~A~~ To demonstrate Autosubmit's new WRROC-based functionality to generate structured provenance data, a workflow was created using an example Autosubmit Project ~~[96]~~ designed using UFZ's mHM (mesoscale Hydrological Model) [97, 98]~~.~~ ~~This workflow was used to validate the RO-Crate produced by Autosubmit. This validation was performed by the Workflow Run RO-Crate community in a public GitHub repository () and also using the aforementioned Runcrate~~, and it was executed with Autosubmit. The resulting Workflow Run Crate is available from Zenodo [96].

3.8 ~~Summary of implementations~~

~~Table 1 shows an overview of the different implementations presented in this section.~~

Table 1. Workflow Run Crate implementations

Impl.	Profile	Version URL/DOI	Example
runcrate	Provenance	[61]	[99]
Galaxy	Workflow	[100]	[66]
COMPSs	Workflow	[74]	[73]
Streamflow	Provenance	[77]	[101]
WfExS	Workflow	~~[102]~~ [78]	~~[?]~~ [81]
Sapporo	Workflow	[87]	[88]
Autosubmit	Workflow	[94]	[96]

Summary of each WRROC implementation, together with the ~~profiles~~ profile it implements, the ~~latest software citation~~ software version that makes it available and an example ~~crate of its application~~RO-Crate. Runcrate is a toolkit that converts CWLProv ROs to Provenance Run Crates, while the others are ~~WMS~~workflow management systems.

4 Exemplary ~~Use Cases~~use cases

We illustrate Workflow Run RO-Crate on two exemplary use cases~~, which~~. These are similar in terms of application domain~~ – machine learning-aided tumour detection in~~, as they both relate to the application of machine learning techniques for the analysis of human prostate images ~~– but~~ for the purpose of supporting cancer tissue detection. However, the use cases are quite different in the way computations are executed and provenance is represented: in the first, the analysis is conducted by means of a CWL workflow and the outcome is represented with Provenance Run Crate; in the second, ~~a combination of~~ Process Run Crate ~~and CPM RO-Crate is used~~ is used in combination with a complementary model to represent a ~~sequence of computations linked to their corresponding CPM provenance information~~provenance chain that can extend beyond the computational analysis.

4.1 Provenance Run Crate for ~~Digital Pathology~~digital pathology

~~We~~ In this section, we present a use case that demonstrates the effectiveness of ~~our most detailed profile~~ the Provenance Run Crate ~~at recording~~ profile at capturing provenance data in the context of digital pathology. More specifically, we demonstrate the generation of RO-Crates to save provenance data associated with the computational annotation of magnified prostate tissue areas and cancer subregions using deep learning models [103]. The image annotation process is implemented in a CWL workflow consisting of three steps, each executing inference on an image using a deep learning model: i) inference of a low-resolution tissue mask to select areas for further processing; ii) high-resolution tissue inference ~~on areas identified in the previous step~~ to refine borders; iii) high-resolution cancer ~~identification on areas identified in the first step.~~ tissue identification. The two tissue inference steps run the same tool, but set different values for the parameter that controls the magnification level, and the second runs on a subset of the image area. The workflow is integrated in the CRS4 Digital Pathology Platform~~()~~ [104], a web-based platform to support clinical studies involving the examination and/or the annotation of digital pathology images.

To assess the interoperability of WRROC, we recorded the provenance of the execution of the same exemplary workflow ~~in two different execution platforms~~on two different WMSs. In the first case, ~~the workflow was executed with the StreamFlow WMS, for which the Provenance Run Crate implementation is discussed in Section 3.4. In the second case,~~ we executed the CWL workflow with cwltool and converted the resulting CWLProv RO to a Provenance Run Crate with the runcrate tool (Section 3.1).

In the second case, the workflow was executed with the StreamFlow WMS (Section 3.4). The RO-Crates obtained in the two cases ~~[99,101]~~ [99,101] are very similar to each other, differing only in a few details~~: for instance, [101] includes the StreamFlow configuration file~~. For instance, Streamflow includes its configuration file in the crate and has separate files for the workflow and the two tools, while ~~[99] has~~ cwltool with runcrate results in the workflow and the tools being stored in a single file (CWL's "packed" format). Apart from these minor differences, the description of the computation is essentially the same~~.~~, so the RO-Crates are fully interoperable. Four actions are represented: the workflow itself, the two executions of the tissue extraction tool and the execution of the tumour classification tool. Each action is linked to the corresponding workflow or tool via the ~~*instrument*~~ *s:instrument* property, and reports its starting and ending time. For each action, input and output slots are referenced by the workflow, while the corresponding values are referenced by the action itself. The data ~~entities and *PropertyValue* instances~~ and *s:PropertyValue* entities corresponding to the input and output values link to the corresponding parameter slots via the ~~*exampleOfWork*~~ *s:exampleOfWork* property, providing information on the values taken by the parameters ~~. The listing below (Fig 1)~~ during execution. Listing 1 shows the output of the `runcrate report` command for the StreamFlow RO-Crate. For each action (workflow or tool run), ~~the tool~~ runcrate reports the associated instrument (workflow or tool), the starting and ending time and the list of inputs and outputs, with ~~arrows pointing~~ pointers from the formal parameter to the corresponding actual value taken during the execution of the action.

~~The *exampleOfWork*~~ The *s:exampleOfWork* link between input / output values and parameter slots is used by `runcrate run` to reconstruct the CWL input ~~parameters document~~ parameter mapping needed to rerun the computation. The ~~*alternateName*~~ *s:alternateName* property (a Schema.org property applicable to all entities), which records the original name of data entities (at the time the computation was run), is also crucial for reproducibility in this case: both StreamFlow and CWLProv, to avoid clashes, record input and output files and directories using their SHA1 checksum as

Listing 1. Output of the `runcrate report` command executed on the Provenance Run Crate generated by StreamFlow in the digital pathology inference use case (Section 4.1). This informal listing of relevant RO-Crate entities describes each step of the execution. Note that inputs and outputs are of different types (not shown): e.g., `tissue_low>0.9` is a string parameter, `6b15de…` is a filename, and `#af0253…` is a collection.

```
\DIFaddendFL action: #30a65cba-1b75-47dc-ad47-1d33819cf156
  instrument: predictions.cwl (['SoftwareSourceCode',
       'ComputationalWorkflow', 'HowTo', 'File'])
  started: 2023-05-09T05:10:53.937305+00:00
  ended: 2023-05-09T05:11:07.521396+00:00
  inputs:
    #af0253d688f3409a2c6d24bf6b35df7c4e271292 <- predictions.cwl#slide
    tissue_low <- predictions.cwl#tissue-low-label
    9 <- predictions.cwl#tissue-low-level
    tissue_low>0.9 <- predictions.cwl#tissue-high-filter
    tissue_high <- predictions.cwl#tissue-high-label
    4 <- predictions.cwl#tissue-high-level
    tissue_low>0.99 <- predictions.cwl#tumor-filter
    tumor <- predictions.cwl#tumor-label
    1 <- predictions.cwl#tumor-level
  outputs:
    06133ec5f8973ec3cc5281e5df56421c3228c221 <- predictions.cwl#tissue
    4fd6110ee3c544182027f82ffe84b5ae7db5fb81 <- predictions.cwl#tumor
action: #457c80d0-75e8-46d6-bada-b3fe82ea0ef1
  step: predictions.cwl#extract-tissue-low
  instrument: extract_tissue.cwl (['SoftwareApplication', 'File'])
  started: 2023-05-09T05:10:55.236742+00:00
  ended: 2023-05-09T05:10:55.910025+00:00
  inputs:
    tissue_low <- extract_tissue.cwl#label
    9 <- extract_tissue.cwl#level
    #af0253d688f3409a2c6d24bf6b35df7c4e271292 <- extract_tissue.cwl#src
  outputs:
    6b15de40dd0ee3234062d0f261c77575a60de0f2 <- extract_tissue.cwl#tissue
action: #d09a8355-1a14-4ea4-b00b-122e010e5cc9
  step: predictions.cwl#extract-tissue-high
  instrument: extract_tissue.cwl (['SoftwareApplication', 'File'])
  started: 2023-05-09T05:10:58.417760+00:00
  ended: 2023-05-09T05:11:03.153912+00:00
  inputs:
    tissue_low>0.9 <- extract_tissue.cwl#filter
    6b15de40dd0ee3234062d0f261c77575a60de0f2 <- extract_tissue.cwl#filter_slide
    tissue_high <- extract_tissue.cwl#label
    4 <- extract_tissue.cwl#level
    #af0253d688f3409a2c6d24bf6b35df7c4e271292 <- extract_tissue.cwl#src
  outputs:
    06133ec5f8973ec3cc5281e5df56421c3228c221 <- extract_tissue.cwl#tissue
action: #ae2163a8-1a2a-4d78-9c81-caad76a72e47
  step: predictions.cwl#classify-tumor
  instrument: classify_tumor.cwl (['SoftwareApplication', 'File'])
  started: 2023-05-09T05:10:58.420654+00:00
  ended: 2023-05-09T05:11:06.708344+00:00
  inputs:
    tissue_low>0.99 <- classify_tumor.cwl#filter
    6b15de40dd0ee3234062d0f261c77575a60de0f2 <- classify_tumor.cwl#filter_slide
    tumor <- classify_tumor.cwl#label
    1 <- classify_tumor.cwl#level
    #af0253d688f3409a2c6d24bf6b35df7c4e271292 <- classify_tumor.cwl#src
  outputs:
    4fd6110ee3c544182027f82ffe84b5ae7db5fb81 <- classify_tumor.cwl#tumor
\DIFdelbeginFL


{%DIFAUXCMD

\texttt{\DIFdelFL{runcrate report}} %DIFAUXCMD
\DIFdelFL{command line output. This informal listing of relevant RO-Crate entities describe each st
\DIFdelFL{is a string parameter, }\texttt{\DIFdelFL{6b15de\ldots}} %DIFAUXCMD
\DIFdelFL{is a filename, and }\texttt{\DIFdelFL{\#af0253\ldots}} %DIFAUXCMD
\DIFdelFL{is a collection.}}
%DIFAUXCMD


\DIFdelendFL \DIFaddbeginFL
```

their names. However, for this particular workflow file names are important: it expects the input ~~dataset~~ image data to be in the MIRAX~~()~~ [106] format, where the “main” dataset file taken as an input parameter by the processing application must be accompanied by a directory of additional data files, in the same location and with the same name, apart from the extension. The runcrate tool uses the ~~*alternateName*~~ *s:alternateName* to rename the input dataset as required, so that the expected pattern can be picked up by the workflow during the re-execution. This use case was the main motivation to include a recommendation to use ~~*alternateName*~~ *s:alternateName* with the above semantics in Process Run Crate.

Thanks to the fact that both RO-Crates were generated following the best practices to support reproducibility mentioned in the profiles, we were able to automatically re-execute both computations with the runcrate tool. This was also made possible by the fact that the CWL workflow included information on which container images to use for each tool. Overall, this shows how reproducibility is a hard-to-achieve goal that can only be supported, but not ensured, by the profiles, since it also depends on factors like the characteristics of the computation, the choice of workflow language and whether best practices such as containerisation are followed.

This use case highlighted the need to add specifications on how to represent multi-file datasets~~[?, section Representing multi-file objects]. In the MIRAX format, in fact, the “main” file must be accompanied by a directory in the same location containing additional files with a specific structure~~ [53, section “Representing multi-file objects”], driven by the need to handle the aforementioned MIRAX image format. To represent ~~this~~these, we added specifications to the Process Run Crate profile on describing ~~“composite”~~ “composite” datasets consisting of multiple files and directories to be treated as a single unit – as opposed to more conventional input or output parameters consisting of a single file. The profile specifies that such datasets should be represented by a ~~*Collection* entity~~ *s:Collection* class linking to individual files and directories via the ~~*hasPart*~~ *s:hasPart* property, and referencing the main part (if any) via the ~~*mainEntity*~~ *s:mainEntity* property. Note that, by adding this specification to Process Run Crate, we also made it available to Workflow Run Crate and Provenance Run Crate. In the output of the runcrate report tool the additional files are not shown, since the formal parameter points to the ~~*Collection* entity~~ *s:Collection* class that describes the whole dataset.

This use case also demonstrates the usage of parameter connections (described in Section 2.3). The RO-Crate resulting from the workflow run contains a representation of all connections between workflow-level parameters (the overall input and output parameters) and tool-level parameters. This allows crate consumers to programmatically find which tool is affected by a workflow-level parameter, thus providing insight on how the workflow works internally (the main feature of the Provenance Run Crate profile). For instance, the `tissue-high-level` workflow parameter is connected to the `level` parameter of the `extract_tissue.cwl` tool by the `extract-tissue-high` step. This parameter regulates the resolution level (pyramidal images are organised into multiple levels of resolution) at which the image is processed in the high-resolution tissue extraction phase. A similar connection is present for the tissue extraction at low resolution. Since *wfrun:ParameterConnections* are referenced from the relevant *s:HowToStep*, the crate consumer can easily determine the resolution level used for both image processing phases from the retrospective provenance.

4.2 Process Run Crate and CPM RO-Crate for cancer detection

This section presents an RO-Crate created to describe an execution of a computational pipeline that trains AI models to detect the presence of carcinoma cells in high-resolution digital images of magnified human prostate tissue. ~~The~~ This RO-Crate makes use of Process Run Crate and CPM RO-Crate~~()~~ [107], an RO-Crate profile that supports the representation of entities described according to the Common Provenance Model (CPM)~~ [108, 109]. The CPM, an~~ [108, 109, 111].

The CPM is a recently developed extension of the W3C PROV model~~ [1] is a recently developed provenance model that~~ [1]. It enables the representation of distributed provenance~~. Distributed provenance~~, which is created when an object involved in the research process ~~,~~ – either digital or physical (e.g., biological material) ~~,~~ – is exchanged between organisations, so that each organisation can document only a portion of the object's life cycle. ~~Individual provenance components are generated, stored, and managed individually by each organisation, and are~~ Using CPM, each involed organisation can document its portion of the life cycle by generating, storing, and managing individual provenance components, which are then linked together in a chain that spans multiple organizations. The CPM prescribes how to represent such provenance, and how to enable its traversal and processing using a common algorithm, independently from the type of object being described. In addition, the CPM defines a notion of meta-provenance, which contains metadata about the history of individual provenance components.

CPM RO-Crate supports the identification of CPM-based provenance and meta-provenance files within an RO-Crate, ~~allowing to pack~~ so that data, metadata, and CPM-based provenance information can be packed together. An RO-Crate generated according to the CPM-RO-Crate profile embeds parts of the distributed provenance, which may be linked to the provenance of precursors and successors of the packed data. The CPM-RO-Crate profile synergises well with Process Run Crate, since the former can add references to CPM-based provenance descriptions of computational executions described with the latter, integrating them in the distributed provenance. Since CPM-based provenance and meta-provenance files are typically themselves produced by computations, Process Run Crate allows to represent these along with the main computations that produce the datasets being exchanged, providing the full picture in a cohesive ensemble.

The use case pipeline consists of three main computational steps: i) a preprocessing step that splits input images into small patches and divides them into a training and a testing set; ii) a training step that trains the model to recognise the presence of carcinoma cells in the images; iii) an evaluation step that measures the accuracy of the trained model on the testing set. In addition to ~~the~~ these pipeline steps, the RO-Crate describes additional computations related to the generation of the CPM provenance and meta-provenance files. All computations are described according to the Process Run Crate profile, while the CPM files are referenced according to the CPM RO-Crate profile. Also represented via Process Run Crate are: the input dataset; the results of the pipeline execution; the scripts that implement the pipeline; the log files generated by the scripts; a script that converts the logs into the CPM files. This approach allowed us to describe all ~~involved~~ elements as a single ~~aggregate, with entities and their relationships represented according to the~~ RO-Crate, which is available on Zenodo [110].

Listing 2 presents the `runcrate report` output for the RO-Crate~~model. The RO-Crate discussed here is available from Zenodo [110].~~, including action inputs and outputs while omitting other details. The listing shows the connections between the actions, forming an "implicit workflow" as discussed in Section 2.1. For instance, the `prov_train.log` file is both an output of the training action (`#train_script:ROCRATE-PUB-…`) and an input of the CPM provenance generation action for the training phase (`#train_script:6efa9a06-…:CPM-provgen`), highlighting the interdependency between the steps.

The CPM files complement the RO-Crate with ~~internal~~ details about the pipeline execution process, such as how the input dataset was split into training and testing sets,

Listing 2. Excerpt of the output of the `runcrate report` command for the AI model training Process Run Crate; only inputs and outputs of the actions are shown. The listing shows the connections between the pipeline actions through the entities they produce or consume – e.g., `cam16_mrxs.h5` is output of the conversion script `convert_script:ff67…` and input for the training script `train_script:ROCRATE…`

```
action: #convert_script:ff67ce65-736f-46d5-9fec-10953cad8695
  inputs:
    wsi/test/
    wsi/train/
    prov_converter_config.json
  outputs:
    cam16_mrxs.h5
    prov_preprocess.log

action: #test_script:ROCRATE-PUB-1438b57a750ce887d4433d9e
  inputs:
    prov_test_config.json
    cam16_mrxs.h5
  outputs:
    predictions.h5
    prov_test.log

action: #test_script:d3cfd9cf-6851-43c6-bee9-c8dc18f22368:CPM-provgen
  inputs:
    prov_test.log
  outputs:
    prov_test.provn
    prov_test.provn.log
    prov_test.png

action: #train_script:ROCRATE-PUB-1438b57a750ce887d4433d9e
  inputs:
    prov_train_config.json
    cam16_mrxs.h5
  outputs:
    prov_train.log
    model/weights/auc_01.ckpt.index
    model/weights/auc_01.ckpt.data-00000-of-00001
    model/weights/auc_02.ckpt.index
    model/weights/auc_02.ckpt.data-00000-of-00001
    model/weights/best_loss.ckpt.index
    model/weights/best_loss.ckpt.data-00000-of-00001
    model/weights/auc_03.ckpt.index
    model/weights/auc_03.ckpt.data-00000-of-00001

action: #train_script:6efa9a06-b8e9-4cfc-88c7-e9d35e5263c3:CPM-provgen
  inputs:
    prov_train.log
  outputs:
    prov_train.provn
    prov_train.png
    prov_train.provn.log

action: #convert_script:9d030b68-70d8-4526-82fe-160d9cfe4806:CPM-provgen
  inputs:
    prov_preprocess.log
  outputs:
    prov_preprocess.provn
    prov_preprocess.png
    prov_preprocess.provn.log

action: #meta_provn_script:86bae258-4c51-4215-854b-32cb49f239ab:CPM-provgen
  inputs:
    prov_train.provn.log
    prov_test.provn.log
    prov_preprocess.provn.log
  outputs:
    meta_provenance.provn
    meta_provenance.png
    meta_provenance.provn.log
```

or detailed information about each training iteration of the AI model. For instance, ~~it~~ the RO-Crate contains a representation of a checkpoint of the AI model after the second training iteration~~. The~~ , with the corresponding entity's attributes ~~contain~~ containing paths to the respective model stored as a file. The entity is related to the respective training iteration activity, which contains the iteration parameters represented as an attribute list. In addition, the CPM generally provides means to link the input dataset provenance to the provenance of its precursors – human prostate tissues and biological samples the tissues were derived from; this is not included in the example because we used a publicly available input database for which provenance of the precursors was not available. However, the linking mechanism for provenance precursors is exactly the same as between the bundles for the AI pipeline parts. While the RO-Crate is focused on the execution of the pipeline, the provenance included in the CPM files intends to be interlinked with provenance of the precursors or successors, providing means to traverse the whole provenance chain. For the described digital pathology pipeline, the precursors would be: ~~(1)~~ i) a biological sample acquired from a patient; ~~(2)~~ ii) slices of the sample processed and put on glass slides; ~~(3)~~ iii) the images created as a result of scanning the slides using a microscope. As a result, combining the CPM and RO-Crate enables the lookup of research artefacts related to the computation across heterogeneous organisations using the underlying provenance chain.

5 Discussion

The RO-Crate profiles presented ~~here~~ in this work provide a unified data model to describe the prospective and retrospective provenance of the execution of a computational workflow, together with contextual metadata about the workflow itself and its associated entities (inputs, outputs, code, etc.). The profiles are flexible, allowing one to tailor the provenance description to a broad variety of use cases, agnostic ~~with respect~~ to the WMS used, and allow describing provenance traces at different levels of granularity. ~~This facilitates developing implementations by multiple workflow systems(often with heterogeneous assumptions and requirements) – six of which have already been developed and are~~ These characteristics facilitate implementing support in workflow systems. Six WMS have already integrated support for a WRROC profile, as described in Section~~3 – allowing to perform comparisons between runs across~~ 3. These new RO-Crate profiles enable interoperability between implementations, which has been demonstrated through the comparison of workflow executions on heterogeneous systems. ~~For instance~~

Choosing to base our approach on the RO-Crate model has led to a number of benefits. The collected provenance data can be treated with standard RDF tools. As an example, the following SPARQL~~()~~ [112] query returns all actions in a Workflow Run RO-Crate, together with their instruments and their starting and ending times~~:~~ , independently of the original workflow type or the WMS that executed the workflow:

```
PREFIX schema: <http://schema.org/>
PREFIX schema: <https://schema.org/>
SELECT ?action ?instrument ?start ?end
WHERE {
  ?action a schema:CreateAction .
  ?action schema:instrument ?instrument .
  OPTIONAL { ?action schema:startTime ?start } .
  OPTIONAL { ?action schema:endTime ?end }
}
```

~~Additionally~~Further, having workflow runs and plans described according to the RO-Crate model allows capturing the context of the workflow itself (e.g. authors, related publications, other workflows, etc.)~~rather than~~, in addition to the trace alone. ~~Being based on RO-Crate, the profiles and their implementations are part of a growing ecosystem of tools and services maintained by the RO-Crate community ().~~

Another advantage of RO-Crate is that the files corresponding to the data entities (inputs, outputs, code, etc.) do not necessarily have to be stored together with the metadata file: for instance, they can be remote and referred to via an http(s) URI. This aspect is mostly relevant in situations where the file is very large or cannot be shared publicly~~: the data entity's identifier can be a URI that is accessible only through~~, since a URI can reference a resource to which access is limited (e.g., accessible only after authentication, or ~~resolvable only within the boundariesof the generating organisation.~~from specific network boundaries, etc.).

The ~~derivation of Workflow Run Crate from Workflow~~ WRROC profiles are extensions of the base RO-Crate ~~and, in turn, of Provenance Run Crate from Workflow Run Crate makes RO-Crates that conform to these profiles compatible with the WorkflowHub workflow registry, allowing workflow runs to be registered and easily found and shared with other researchers. Additionally, the inheritance mechanism allows reusing the specifications already developed for Workflow~~ specification that specialise it for the use case of workflow execution provenance representation. The additional terms, constraints and recommendations introduced by the profiles allow users to represent classes and relationships involved in a workflow execution in a precise and detailed way, so that consumers of the RO-Crate ~~, which form part of the guidelines on representing the prospective provenance~~ can programmatically retrieve the relevant information according to predefined patterns and act upon it. This is a crucial advantage over using the base RO-Crate specification, which was not designed to answer the competency questions defined for capturing the provenance of workflow executions.

The ~~Workflows Community Summit [113]~~ ability to build FAIR into Workflow Management Systems was identified as one of the current open challenges in the Scientific Workflows domain ~~the ability to build FAIR into Workflow Management Systems~~at the Workflows Community Summit [113], with the objective of achieving FAIR Computational Workflows. The profiles introduced in this article ~~are able to~~ help tackle this challenge by introducing interoperable metadata among WMSs that captures the provenance of their corresponding workflow executions. The derivation of Workflow Run Crate, and in turn Provenance Run Crate, from Workflow RO-Crate makes the digital objects that conform to these new profiles compatible with the WorkflowHub workflow registry [40]. This design entails that Workflow Run RO-Crates directly reference the workflow with which the provenance was generated, and it allows workflow runs to be registered on WorkflowHub and easily found and shared with other researchers. Additionally, the inheritance mechanism allows reusing the specifications already developed for Workflow RO-Crate, which form part of the guidelines on representing the prospective provenance.

The Workflow Run RO-Crate profiles, the associated tooling, the implementations and the examples are developed ~~by a community that runs regular virtual meetings (every two weeks at the~~ and supported by the open WRROC Community. At the time of writing~~) and coordinates on Slack and the RO-Crate mailing list. The WRROC community~~, the Community numbers nearly 40 members and brings together members of the RO-Crate community [37], WMS users and developers, ~~Workflow~~ workflow users and developers, GA4GH [114] Cloud developers and provenance model authors, and is open to anyone who is interested in the representation of workflow execution provenance. The inclusion of WMS developers and workflow users ~~was~~ has been key to keeping the specifications flexible, easy to implement and grounded on real use cases, while the diversity of the stakeholders ~~allowed to keep~~ has included a plurality of viewpoints while driving the model's development forward~~.~~

~~One of the main benefits of this development process is that the profiles are already in use, with seven implementations (six WMS and one conversion tool) already available~~ , resulting in profiles that are already being used (as described in ~~section 3.~~ Section 3).

In the following subsections, we provide an evaluation of the metadata coverage of runcrate and we discuss how WRROC relates to standards such as W3C ~~PROV~~ PROV-O and to other community projects.

5.1 Evaluation of metadata coverage using runcrate convert

~~In order to assess the metadata coverage of runcrate, we performed a qualitative analysis of the tool's *convert* mode, in which we evaluated how the generated RO-Crates preserve the~~ Since CWLProv was a starting point in the development of WRROC (Section 3.1), as a baseline validation we chose to verify that the metadata contained in ~~the CWLProv ROs from which they are derived. For this analysis, we followed the same approach as for an earlier evaluation of CWLProv [115]. In that work , we identified and analysed three levels of representation: firstly, in RDF; secondly, in a structured, but CWL-specific document; and finally, in an unstructured, human readable format. From this earlier analysis~~ CWLProv ROs is preserved in the RO-Crates produced by their conversion through runcrate's *convert* command. In previous work we had conducted a qualitative analysis of metadata coverage in CWLProv (version 0.6.0), based on concrete examples of ROs associated with a realistic bioinformatics workflow [115]; in this work we repeated this analysis for WRROC, and compared the WRROC RDF representation (in `ro-crate-metadata.json`) with the CWLProv RDF provenance graph. To summarise, the analysis focuses on the comparison of the degree of representation by the two models of six provenance data types defined in [115], which we recall here for clarity.

T1. **Scientific context**: the choices which were made in the design of the workflow and parameter values.

T2. **Data**: input and output data.

T3. **Software**: the tools directly orchestrated by the workflow, ~~we concluded that the CWLProv RDF representation of the workflow runs lacked many provenance metadata that was included inCWL-specific documents, such as the packed workflow and input parameter file. For example, the CWLProv RDF only contained the name of each workflowstep, without including the link to the underlying CommandLineTool or nested Workflow that was executed; information that could be extracted from the packed 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.

Each type is in turn articulated in a set of data subtypes, forming a hierarchy of elements that should be represented in workflow provenance data to satisfy a range of use cases spanning from supporting workflow development to supporting a service based on the execution of the workflow, with several other use cases in between. For a full motivation and description of the criteria the reader may refer to the original work [115].

~~In our analysis of runcrate, we compared the CWLProv RDF provenance graph with~~ Our analysis shows that, overall, most of the information contained in the CWLProv RDF is transferred to the RO-Crate metadata~~file~~. The results ~~of the analysis~~ are summarised in Table~~2. The three dots (...) in the WRROC column indicate that the concept is supported in an RO-Crate using existing schema.org vocabulary (e.g.)but is not required or recommended by the WRROC profiles. Overall, most of the information contained in CWLProv RDF is transferred to the RO-Crate metadata. In addition, the~~ 2; for completeness, we also report the (non-RDF) representation of provenance metadata in CWL-specific documents (`packed.cwl` and `primary-job.json`), which are included in both CWLProv ROs and RO-Crates generated by runcrate. We observe that out of the total 20 provenance data subtypes that are part of the analysis, WRROC represented 13 (65%) of them (9 fully, 4 partially), while CWLProv RDF captured 8 (3 fully, 5 partially). The representation of some entire categories of metadata has improved ~~,~~ notably Workflow parameters (WF2), which were insufficiently described in CWLProv RDF, but defined with type and format in RO-Crate. Moreover, the ~~format of input files (D2), which was partially represented in CWLProvRDF, is fully represented in~~ Workflow Run RO-Crate RDF contains a representation of tools orchestrated by the workflow (T3), as well as a much more extensive description of the workflow itself (T4) compared to CWLProv.

In conclusion, our analysis shows that runcrate preserves most provenance metadata previously shown to be relevant in realistic RO use case scenarios. ~~The full~~ More detailed results of the analysis can be found in ~~[?].~~ [116].

~~From this analysis it is worth highlighting the gaps and potential for Workflow Run RO-Crates. Several areas have been flagged by this study as important aspects of workflow metadata, such as Data Access (D3), Software Documentation (SW2) and Workflow Requirements (WF3). Many such aspects require human annotation and cannot be provided by workflow engines alone, although they may be propagated from workflow and tool definitions. Some areas like Consumed Resources (EX2) require additional terms to be defined, and are part of future work.~~

5.2 Workflow Run RO-Crate and the W3C PROV standard

~~Our aim is to be~~ One of our aims for the WRROC profiles is to make them compatible with both Schema.org and W3C PROV. Provenance Run Crate is the profile that most closely matches the level of detail provided by CWLProv, which extends W3C PROV. Table 3 shows how the main ~~entities~~ classes and relationships represented by Provenance Run Crate map to PROV constructs, using the SKOS vocabulary to indicate the type of relationship between each pair of terms. A machine-readable version of the mapping can be found in the RO-Crate accompanying this article ~~[117] ().~~ [117, 118].

5.3 Five Safes Workflow Run Crate

The *Five Safes RO-Crate* [119] profile has been developed to extend the Workflow Run ~~RO-~~Crate profile for use in Trusted Research Environments (TRE)~~in order to~~, following the Five Safes Framework [121] to better handle sensitive health data in federated workflow execution across TREs in the UK ~~[120] and following the Five Safes Framework [121]~~ [120]. A crate with a workflow run request references a pre-approved workflow and project details for manual and automated assessment according to the TRE's agreement policy for the sensitive dataset.

The crate then goes through multiple phases internal to the TRE, including validation, sign-off, workflow execution and disclosure control. At this stage the crate is also conforming to the Workflow Run Crate profile. The final crate is then safe to be made public.

This extension of Workflow Run Crate documents and supports the *human review process* – important for transparency on TRE data usage. The initial implementation of this ~~profile~~ process used WfExS as the workflow execution backend, and this approach will form the basis for further work on implementing federated workflow execution in the British initiatives DARE UK and HDR UK [122] and in the European EOSC-ENTRUST project for Trusted Research Environments~~().~~ [123].

5.4 Biocompute Object RO-Crate

IEEE 2791-2020 [124], colloquially known as *Biocompute Objects* (BCO), is a standard for representing provenance of a genomic sequencing pipeline, intended for submission of the workflow to regulatory bodies ~~,~~ — e.g. as part of a personalised medical treatment method [125]. The BCO is represented as a single JSON file which includes description of the workflow and its steps and intended purpose, as well as references for tools used and data sources accessed. There is overlap in the goals of BCO and Workflow Run Crate profiles, however their intentions and focus are different. BCO is primarily conveying a computational method for the purpose of manual regulatory review and further reuse, with any values provided as an exemplar run. A Workflow Run Crate~~however~~, however, is primarily documenting a particular workflow execution, and the workflow is associated to facilitate rerun rather than reuse.

Previously, a guide to packaging BioCompute Objects using RO-Crate~~()~~ [126] was developed as a profile to combine both standards [127]. In this early approach, RO-Crate was primarily a vessel to transport the BCO along with its constituent resources, including the workflow and data files, as well as to provide these resources with additional typing and licence metadata that is not captured by the BCO JSON. Further work is being planned with the BCO community to update the BCO-RO profile to align with the newer Workflow Run ~~Crate~~ RO-Crate profiles.

6 Conclusion and future work

~~In this work we presented~~ The Workflow Run RO-Crate ~~, a collection of RO-Crate profiles to represent~~ profile collection presented in this manuscript is a new model to represent and package both the prospective and the retrospective provenance relating to the ~~provenance of the~~ execution of computational workflows ~~at different levels of granularity. We described each profile and their corresponding implementations, shown how they apply to real use cases and described the community behind their development process.Workflow Run RO-Crate~~ in a way that is machine-actionable, interoperable, independent of the specific workflow language or execution system, and including support for re-execution. These new profiles build on RO-Crate and Schema.org to include contextual information and bundle together all objects of the workflow execution (inputs, outputs, code, etc.). Our approach minimizes the set of mandatory metadata items and defines a hierarchy of profiles – Process Run Crate, Workflow Run Crate, and Provenance Run Crate – that capture provenance information at increasing levels of detail and complexity. This flexible approach increases the model's adaptability to the diverse landscape of WMSs used in practice, and modulates the implementation effort as a function of the requirements of the specific use case. As a result, there has already been ~~adopted by~~ significant uptake of Workflow Run RO-Crate, as shown by its adoption in six WMS, including Galaxy, StreamFlow and COMPSs~~. The flexibility of our model eases its implementation in more systems, allowing interoperability between their workflow run descriptions. ~~; in addition, the `runcrate` toolkit has been implemented as part of this work providing various inspection, conversion and re-execution functionalities. Moreover, we have shown how WRROC has been applied in real use cases.

Workflow Run RO-Crate is an ongoing project~~driven by an open community. A natural consequence of this is that the profiles ~~. Therefore, our profiles and the surrounding software are not static entities, but keep being updated to cater for new requirements and use cases. ~~In-progress features are tracked in the GitHub repository issues section () and are open to discussion for the community. New features under discussion include a representation of the execution environment and recording workflow resource usage. The runcrate toolkit is planned to be expanded both to ~~As examples of ongoing work, at the time of writing there are plans to expand the runcrate toolkit to better support the ~~current features and to include new ones that may arise.~~

~~Many of the presented implementations ~~creation and querying of WRROC objects. Also, work is ongoing to implement automated conformance validation of crates. In addition, several of the implementations presented in this work will also develop new features. For ~~example~~instance, the Galaxy ~~implementation will add ~~community plans to extend its WRROC support to: include metadata detailing each step of a workflow run to conform to the Provenance Run Crate profile; develop and/or integrate RO-Crate more deeply with import and export of Galaxy histories~~through the implementation of a profile~~; and further ~~developing features to allow for ~~develop user-guided import of RO-Crates as Galaxy datasets, histories and workflows.

~~Finally~~Further, we are currently exploring the cloud execution of Workflow Run RO-Crates. ~~On the one hand, the ~~The Workflow Execution Service (WES) specification is used by the Global Alliance for Genomics and Health (GA4GH) [114] to enable WMS-agnostic interpretation of workflows and scheduling of task execution. ~~On the other hand~~In addition, the Task Execution Service (TES) specification enables the execution of individual, atomic, ~~containerized ~~containerised tasks in a compute backend-independent manner.

We are planning to undertake an in-depth analysis of the degree of interoperability between the TES and WES API standards – roughly the equivalents of Process and Workflow Run Crates, respectively – by placing their focus on the actual execution of tasks/processes and workflows in cloud environments and liaising with the GA4GH Cloud community to align schemas where necessary. We will then build an interconversion library that attempts to ~~(1) ~~i) construct WES workflow and TES task run requests from RO-Crates containing Provenance, Workflow or Process Run requests and therefore allow their easy (re)execution on any GA4GH Cloud API-powered infrastructure, and ~~(2)~~ ii) bundle information from the WES and TES (as well as other GA4GH Cloud API resources, where available) to create or extend RO-Crates with standards-compliant Process, Workflow or even Provenance RO-Crates.

~~Supporting information~~

~~Process Run Crate profile [?] Workflow Run Crate profile [?]Provenance Run Crate profile [?]Machine-readable mapping from WRROC to PROV [117] Workflow Run~~ The maintenance and development of WRROC is driven by an open community, currently numbering about 40 members. The Community runs regular virtual meetings (every two weeks at the time of writing) and coordinates on Slack and the RO-Crate ~~Introduction [128] (from Galaxy Smörgåsbord 2023) WRROC implementations and examples (see Table 1) ~~mailing list. Naturally, feedback and contributions from the community are welcome and encouraged, and new requirements and features are discussed and sustained, particularly through the WRROC GitHub repository issue tracker [48]. Through the open Community we expect to encourage and support further adoption of WRROC, be it by the other WMS or other use cases, maybe in time converging towards a common workflow execution provenance representation.

Acknowledgments

The authors would like to thank all participants to the Workflow Run RO-Crate working group meetings for the fruitful discussions and valuable feedback.

Author contributions

Author contributions following the CRediT Taxonomy:

Simone Leo Conceptualization, Data Curation, Investigation, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing – Original Draft preparation, Writing – Review & Editing

Michael R. Crusoe Conceptualization, Investigation, Software, Supervision

Laura Rodríguez-Navas Software, Writing – Original Draft preparation

Raül Sirvent Data Curation, Software, Writing – Original Draft preparation, Writing – Review & Editing

Alexander Kanitz Writing – Original Draft preparation, Writing – Review & Editing

Paul De Geest Data Curation, Software, Writing – Original Draft preparation

Rudolf Wittner Data Curation, Writing – Original Draft preparation, Writing – Review & Editing

Luca Pireddu Funding acquisition, Project Administration, Supervision, Writing – Review & Editing

Daniel Garijo Conceptualization, Formal Analysis, Writing – Original Draft preparation, Writing – Review & Editing

José M. Fernández Data Curation, Software, Writing – Original Draft preparation

Iacopo Colonnelli Data Curation, Software, Writing – Original Draft preparation

Matej Gallo Data Curation, Software

Tazro Ohta Data Curation, Software, Writing – Original Draft preparation

Hirotaka Suetake Data Curation, Software, Writing – Original Draft preparation

Salvador Capella-Gutierrez Funding Acquisition, Resources, Supervision, Writing – Original Draft preparation

Renske de Wit Software, Writing – Original Draft preparation, Writing – Review & Editing

Bruno de Paula Kinoshita Data Curation, Software, Writing – Original Draft preparation, Writing – Review & Editing

Stian Soiland-Reyes Conceptualization, Formal Analysis, Funding Acquisition, Investigation, Methodology, Resources, Software, Supervision, Visualization, Writing – Original Draft preparation, Writing – Review & Editing

References

1. Moreau L, Missier P, Belhajjame K, B'Far R, Cheney J, Coppens S, et al. PROV-DM: The PROV Data Model. W3C Recommendation 30 April 2013 [cited 2023 Dec 7]. https://www.w3.org/TR/2013/REC-prov-dm-20130430/

2. Herschel M, Diestelkämper R, Ben Lahmar H. A survey on provenance: What for? What form? What from? The VLDB Journal, 2017;26:881–906. doi: 10.1007/s00778-017-0486-1

3. ~~Gauthier J, Vincent AT, Charette SJ, Derome N. A brief history of bioinformatics. Briefings in Bioinformatics, 2019;20(6):1981–1996. doi: 10.1093/bib/bby063~~

4. Himanen L, Geurts A, Foster AS, Rinke P. Data-Driven Materials Science: Status, Challenges, and Perspectives. Advanced Science, 2019;6(21):1900808. doi: 10.1002/advs.201900808

5. Gauthier J, Vincent AT, Charette SJ, Derome N. A brief history of bioinformatics. Briefings in Bioinformatics, 2019;20(6):1981–1996. doi: 10.1093/bib/bby063

6. Huntingford C, Jeffers ES, Bonsall MB, Christensen HM, Lees T, Yang H. Machine learning and artificial intelligence to aid climate change research and preparedness. Environmental Research Letters, 2019;14(12):124007. doi: 10.1088/1748-9326/ab4e55

7. Lebo T, Sahoo S, McGuinness D, Belhajjame K, Cheney J, Corsar D, et al. PROV-O: The PROV Ontology. W3C Recommendation 30 April 2013 [cited 2023 Dec 7]. https://www.w3.org/TR/2013/REC-prov-o-20130430/

8. W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation 11 December 2012 [cited 2023 Dec 7]. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

9. Missier P, Dey S, Belhajjame K, Cuevas-Vicenttín V, Ludäscher B. D-PROV: extending the PROV provenance model with workflow structure. In Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance (TaPP '13), 2013.

10. Cuevas-Vicenttín V, Ludäscher B, Missier P, Belhajjame K, Chirigati F, Wei Y, et al. ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance, 2016 [cited 2023 Dec 7]. https://purl.dataone.org/provone-v1-dev

11. Garijo D, Gil Y. A new approach for publishing workflows: abstractions, standards, and linked data. In Proceedings of the 6th workshop on Workflows in support of large-scale science (WORKS '11) 2011. doi: 10.1145/2110497.2110504

12. Garijo D, Gil Y. Augmenting PROV with Plans in P-PLAN: Scientific Processes as Linked Data. In Proceedings of the Second International Workshop on Linked Science, 2012.

13. Freire J, Koop D, Santos E, Silva CT. Provenance for Computational Tasks: A Survey. Computing in Science & Engineering 2012;10(3):11–21. doi: 10.1109/MCSE.2008.79

14. ~~Garijo D, Gil Y, Corcho O. Towards Workflow Ecosystems through Semantic and Standard Representations. In Proceedings of the 9th Workshop on Workflows in Support of Large-Scale Science 2014. doi: 10.1109/works.2014.13~~

15. Gil Y, Ratnakar V, Kim J, Gonzalez-Calero P, Groth P, Moody J, et al. Wings: Intelligent Workflow-Based Design of Computational Experiments. IEEE Intelligent Systems 2011;26(1). doi: 10.1109/MIS.2010.9

16. ~~Costa F, Silva V, de Oliveira D, Ocaña K, Ogasawara E, Dias J, et al. Capturing and querying workflow runtime provenance with PROV: a practical approach~~

17. Garijo D, Gil Y, Corcho O. Towards Workflow Ecosystems through Semantic and Standard Representations. In Proceedings of the ~~Joint EDBT~~9th Workshop on Workflows in Support of Large-Scale Science 2014. doi: 10.1109/~~ICDT 2013 Workshops 2013. doi: 10.1145/2457317.2457365~~works.2014.13

18. Scheidegger CE, Vo HT, Koop D, Freire J, Silva CT. Querying and re-using workflows with VisTrails. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data 2008. doi: 10.1145/1376616.1376747

19. Costa F, Silva V, de Oliveira D, Ocaña K, Ogasawara E, Dias J, et al. Capturing and querying workflow runtime provenance with PROV: a practical approach. In Proceedings of the Joint EDBT/ICDT 2013 Workshops 2013. doi: 10.1145/2457317.2457365

20. Atkinson M, Gesing S, Montagnat J, Taylor I. Scientific workflows: Past, present and future. Future Generation Computer Systems 2017;75:216–227. doi: 10.1016/j.future.2017.05.041

21. Pérez B, Rubio J, Sáenz-Adán C. A systematic review of provenance systems. Knowledge and Information Systems 2018;57:495–543. doi: 10.1007/s10115-018-1164-3

22. Belhajjame K, Zhao J, Garijo D, Gamble M, Hettne K, Palma R, et al. Using a suite of ontologies for preserving workflow-centric research objects. Journal of Web Semantics 2015;32:16–42. doi: 10.1016/j.websem.2015.01.003

23. Bechhofer S, Buchan I, De Roure D, Missier P, Ainsworth J, Bhagat J, et al. Why linked data is not enough for scientists. Future Generation Computer Systems 2013;29(2):599–611. doi: 10.1016/j.future.2011.08.004

24. Samuel S, König-Ries B. End-to-End provenance representation for the understandability and reproducibility of scientific experiments using a semantic approach. Journal of Biomedical Semantics 2022;13:1. doi: 10.1186/s13326-021-00253-1

25. Samuel S, König-Ries B. ProvBook: Provenance-based Semantic Enrichment of Interactive Notebooks for Reproducibility. The 17th International Semantic Web Conference (ISWC) 2018 Demo Track, 2018.

26. Khan FZ, Soiland-Reyes S, Sinnott RO, Lonie A, Goble C, Crusoe MR. Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. GigaScience 2019;8(11):giz095. doi: 10.1093/gigascience/giz095

27. Chard K, D'Arcy M, Heavner B, Foster I, Kesselman C, Madduri R, et al. I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. 2016 IEEE International Conference on Big Data (Big Data) 2016;319–328. doi: 10.1109/BigData.2016.7840618

28. Soiland-Reyes S, Khan FZ, Crusoe MR. common-workflow-language/cwlprov: CWLProv 0.6.0. Zenodo, 2018. doi: 10.5281/zenodo.1471585

29. Soiland-Reyes S, Alper P, Goble C. Tracking workflow execution with TavernaProv. Zenodo, 2016. doi: 10.5281/zenodo.51314

30. Crusoe MR, Abeln S, Iosup A, Amstutz P, Chilton J, Tijanić N, et al. Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language. Communications of the ACM, 2022;65(6):54–63. doi: 10.1145/3486897

31. Common Workflow Language Implementations [cited 2024 May 24]. `https://www.commonwl.org/implementations/`

32. Soiland-Reyes S. The Roterms ontology. Release 30 July 2015 [cited 2024 May 24]. `https://wf4ever.github.io/ro/2016-01-28/roterms/`

33. Amstutz P, Crusoe MR, Khan FZ, Soiland-Reyes S, Singh M, Kumar K, et al. common-workflow-language/cwltool: 3.1.20230127121939. Zenodo, 2023. doi: 10.5281/zenodo.7575947

34. Lordan F, Tejedor E, Ejarque J, Rafanell R, Álvarez J, Marozzo F, et al. ServiceSs: An interoperable programming framework for the cloud. Journal of Grid Computing 2014;12:67–91. doi: 10.1007/s10723-013-9272-5

35. Research Object Bundle context [cited 2024 May 24] `https://w3id.org/bundle/context`

36. Chard K, Gaffney N, Jones MB, Kowalik K, Ludäscher B, McPhillips T, et al. Application of BagIt-Serialized Research Object Bundles for Packaging and Re-Execution of Computational Analyses. 2019 15th International Conference on eScience (eScience) 2019. doi: 10.1109/eScience.2019.00068

37. Soiland-Reyes S, Sefton P, Crosas M, Castro LJ, Coppens F, Fernández JM, et al. Packaging research artefacts with RO-Crate. Data Science 2022;5(2):97–138. doi: 10.3233/DS-210053

38. Guha RV, Brickley D, Macbeth S. Schema.org: Evolution of Structured Data on the Web: Big data makes common schemas even more necessary. Queue 2015;13(9):10–37. doi: doi:10.1145/2857274.2857276

39. Sporny M, Longley D, Kellogg G, Lanthaler M, Champin PA, Lindström N. JSON-LD 1.1: A JSON-based Serialization for Linked Data. W3C Recommendation 16 July 2020 [cited 2023 Dec 11]. `https://www.w3.org/TR/2020/REC-json-ld11-20200716/`

40. Goble C, Soiland-Reyes S, Bacall F, Owen S, Williams A, Eguinoa I, et al. Implementing FAIR Digital Objects in the EOSC-Life Workflow Collaboratory. Zenodo, 2021. doi: 10.5281/zenodo.4605654

41. Bacall F, Williams AR, Owen S, Soiland-Reyes S. Workflow RO-Crate Profile 1.0. WorkflowHub community, 2022 [cited 2023 Dec 11]. `https://w3id.org/workflowhub/workflow-ro-crate/1.0`

42. Batista D, Gonzalez-Beltran A, Sansone SA, Rocca-Serra P. Machine actionable metadata models. Scientific Data, 2022;9:592. doi: 10.1038/s41597-022-01707-6

43. Isaac A, Summers E. SKOS Simple Knowledge Organization System Primer. W3C Working Group Note 18 August 2009 [cited 2023 Dec 11]. `https://www.w3.org/TR/2009/NOTE-skos-primer-20090818/`

44. Soiland-Reyes S, Sefton P, Castro LJ, Coppens F, Garijo D, Leo S, et al. Creating lightweight FAIR Digital Objects with RO-Crate. Research Ideas and Outcomes, 2022;8:e93937. doi: 10.3897/rio.8.e93937

45. Workflow Run RO-Crate [cited 2024 May 24]. `https://www.researchobject.org/workflow-run-crate`

46. Workflow Run RO-Crate competency questions [cited 2024 May 24]. `https://www.researchobject.org/workflow-run-crate/requirements`

47. SPARQL queries for the Competency Questions [cited 2024 June 4]. `https://github.com/ResearchObject/workflow-run-crate/tree/main/docs/sparql`

48. Workflow Run RO-Crate GitHub repository [cited 2024 July 2]. `https://github.com/ResearchObject/workflow-run-crate`

49. RO-Crate JSON-LD context, version 1.1 [cited 2024 May 24]. `https://www.researchobject.org/ro-crate/1.1/context.jsonld`

50. Gray A, Goble C, Jimenez R, The Bioschemas Community (2017). Bioschemas: From Potato Salad to Protein Annotation. ISWC (Posters, Demos & Industry Tracks), 2017. `https://iswc2017.semanticweb.org/paper-579/`

51. Bioschemas ComputationalWorkflow Profile, version 1.0-RELEASE (09 March 2021) [cited 2024 May 24]. `https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE`

52. ro-terms: Workflow run namespace [cited 2024 Jul 03]. `https://w3id.org/ro/terms/workflow-run`

53. Workflow Run RO-Crate working group. Process Run Crate specification. Version ~~0.4~~0.5. Zenodo, ~~2023. doi: ~~2024. doi: 10.5281/zenodo~~.10203944~~.12158562

54. ~~Meurisse M, Estupiñán-Romero F, González-Galindo J, Martínez-Lizaga N, Royo-Sierra S, Saldner S, et al. Federated causal inference based on real-world observational data sources: application to a SARS-CoV-2 vaccine effectiveness assessment. BMC Medical Research Methodology 2023;23:248. doi: 10.1186/s12874-023-02068-3~~

55. Workflow Run RO-Crate working group. Workflow Run Crate specification. Version ~~0.4~~0.5. Zenodo, ~~2023. doi: ~~2024. doi: 10.5281/zenodo~~.10203971~~.12159311

56. Köster J, Rahmann S. Snakemake–a scalable bioinformatics workflow engine. Bioinformatics 2012;28(19):2520–2522. doi: 10.1093/bioinformatics/bts480

57. Colonnelli I, Cantalupo B, Merelli I, Aldinucci M. StreamFlow: cross-breeding Cloud with HPC. IEEE Transactions on Emerging Topics in Computing, 2021;9(4):1723–1737. doi: 10.1109/TETC.2020.3019202

58. The Galaxy Community. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2022 update. Nucleic Acids Research 2022;50(W1):W345–W351. doi: 10.1093/nar/gkac247

59. Workflow Run RO-Crate working group. Provenance Run Crate specification. Version ~~0.4~~0.5. Zenodo, ~~2023. doi: ~~2024. doi: 10.5281/zenodo~~.10203978~~.12160782

60. Schema.org HowToStep definition [cited 2024 May 24]. `https://schema.org/HowToStep`

61. Leo S, Soiland-Reyes S, Crusoe MR. Runcrate. Version 0.5.0. Zenodo, 2023. doi: 10.5281/zenodo.10203433

62. Blankenberg D, Von Kuster G, Bouvier E, Baker D, Afgan E, Stoler N, et al. Dissemination of scientific software with Galaxy ToolShed. Genome Biology 2014;15:403. doi: 10.1186/gb4161

63. De Geest P, Droesbeke B, Eguinoa I, Gaignard A, Huber S, Kinoshita B, et al. ResearchObject/ro-crate-py: ro-crate-py 0.9.0. Zenodo, 2023. doi: 10.5281/zenodo.10017862

64. De Geest P, Coppens F, Soiland-Reyes S, Eguinoa I, Leo S. Enhancing RDM in Galaxy by integrating RO-Crate. Research Ideas and Outcomes, 2022;8:e95164. doi: 10.3897/rio.8.e95164

65. Galaxy Workflow Format 2 Description [cited 2024 May 24]. `https://galaxyproject.github.io/gxformat2/v19_09.html`

66. De Geest P. Run of an example Galaxy collection workflow. Zenodo, 2023. doi: 10.5281/zenodo.7785861

67. Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM et al. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. Lecture Notes in Computer Science, 2004;3241:97–104. doi: 10.1007/978-3-540-30218-6_19.

68. Dagum L, Menon R. OpenMP: an industry standard API for shared-memory programming. IEEE Computational Science and Engineering 1998;5(1):46-55. doi: 10.1109/99.660313.

69. Lam SK, Pitrou A, Seibert S. Numba: a LLVM-based Python JIT compiler. In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC 2015. doi: 10.1145/2833157.2833162.

70. Sirvent R, Conejero J, Lordan F, Ejarque J, Rodriguez-Navas L, Fernandez JM, et al. Automatic, Efficient, and Scalable Provenance Registration for FAIR HPC Workflows. 2022 IEEE/ACM Workshop on Workflows in Support of Large-Scale Science (WORKS), 2022. doi: 10.1109/works56498.2022.00006

71. MareNostrum 4 user's guide [cited 2024 May 24]. `https://bsc.es/supportkc/docs/MareNostrum4/intro/`

72. Poiata N, Satriano C, Vilotte JP, Bernard P, Obara K. Multiband array detection and location of seismic sources recorded by dense seismic networks. Geophysical Journal International, 2016;205(3):1548–1573. doi: 10.1093/gji/ggw071

73. Poiata N, Satriano C, Conejero J. BackTrackBB: Multi-band array detection and location of seismic sources (PyCOMPSs implementation). Zenodo, 2023. doi: 10.5281/zenodo.7788030

74. Ejarque J, Lordan F, Badia RM, Sirvent R, Lezzi D, Vazquez F, et al. COMPSs. Version v3.2. Zenodo, 2023. doi: 10.5281/zenodo.7975340

75. Reis D, Piedade B, Correia FF, Dias JP, Aguiar A. Developing Docker and Docker-Compose Specifications: A Developers' Survey. IEEE Access, 2022;10:2318–2329. doi: 10.1109/ACCESS.2021.3137671

76. Zerouali A, Opdebeeck R, De Roover C. Helm Charts for Kubernetes Applications: Evolution, Outdatedness and Security Risks. 2023 IEEE/ACM 20th International Conference on Mining Software Repositories, 2023;523–533. doi: 10.1109/MSR59073.2023.00078

77. Colonnelli I, Cantalupo B, Aldinucci M, Saitta G, Mulone A. StreamFlow. Version 0.2.0.dev10. Software Heritage Archive, 2023. `https://identifiers.org/swh:1:rev:b2014add57189900fa5a0a0403b7ae3a384df73b`

78. Fernández ~~González JM. RO-Crate from staged WfExS working directory 047b6dfc-3547-4e09-92f8-df7143038ff4 (overbridging templon). Zenodo~~JM, ~~2023. doi:~~ Rodríguez-Navas L, Muñoz-Cívico A, Iborra P, Lea D. WfExS-backend. Version 1.0.0a0. Zenodo, 2024. doi: 10.5281/zenodo~~.10091550~~.12589121

79. Di Tommaso P, Chatzou M, Floden EW, Prieto Barja P, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nature Biotechnology 2017;35:316–319. doi: 10.1038/nbt.3820

80. Bouyssié D, Altıner P, Capella-Gutierrez S, Fernández JM, Hagemeijer YP, Horvatovich P, et al. WOMBAT-P: Benchmarking Label-Free Proteomics Data Analysis Workflows. Journal of Proteome Research, 2023. doi: 10.1021/acs.jproteome.3c00636

81. Fernández González JM. RO-Crate from staged WfExS working directory 047b6dfc-3547-4e09-92f8-df7143038ff4 (overbridging templon). Zenodo, 2024. doi: 10.5281/zenodo.12588049

82. Fernández JM. RO-Crate from staged WfExS working directory a37fee9e-4288-4a9e-b493-993a867207d0 (meer oxometalate). Zenodo, 2024. doi: 10.5281/zenodo.12622362

83. Suetake H, Tanjo T, Ishii M, Kinoshita BP, Fujino T, Hachiya T, et al. Sapporo: A workflow execution service that encourages the reuse of workflows in various languages

in bioinformatics [version 1; peer review: 2 approved with reservations]. F1000Research 2022;11:889. doi: 10.12688/f1000research.122924.1

84. Vivian J, Rao AA, Nothaft FA, Ketchum C, Armstrong J, Novak A, et al. Toil enables reproducible, open source, big biomedical data analyses. Nature Biotechnology 2017;35(4):314–316. doi: 10.1038/nbt.3772

85. ro-terms: Sapporo namespace [cited 2024 May 28]. https://github.com/ResearchObject/ro-terms/tree/master/sapporo

86. Suetake H, Fukusato T, Igarashi T, Ohta T. A workflow reproducibility scale for automatic validation of biological interpretation results. GigaScience 2023;12:giad031. doi: 10.1093/gigascience/giad031

87. Suetake H, Ohta TI, Tanjo T, Ishii M, Kinoshita BP, DrYak. sapporo-wes/sapporo-service: 1.5.1. Zenodo, 2023. doi: 10.5281/zenodo.10134452

88. Ohta T, Suetake H. Example of Workflow Run RO-Crate Output in Sapporo. Zenodo, 2023. doi: 10.5281/zenodo.10134581

89. Manubens-Gil D, Vegas-Regidor J, Prodhomme C, Mula-Valls O, Doblas-Reyes FJ. Seamless management of ensemble climate prediction experiments on HPC platforms. 2016 International Conference on High Performance Computing & Simulation (HPCS), 2016;895-900. doi: 10.1109/HPCSim.2016.7568429

90. Yoo AB, Jette MA, Grondona M. SLURM: Simple Linux Utility for Resource Management. Job Scheduling Strategies for Parallel Processing (JSSPP 2003). Lecture Notes in Computer Science, 2003;2862. doi: 10.1007/10968987_3

91. Feng H, Misra V, Rubenstein D. PBS: a unified priority-based scheduler. Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, 2007;203–214. doi: 10.1145/1254882.1254906

92. Bahra A. Managing work flows with ecFlow. ECMWF Newsletter, 2011;129:30–32. doi: 10.21957/nr843dob

93. Oliver H, Shin M, Matthews D, Sanders O, Bartholomew S, Clark A, et al. Workflow Automation for Cycling Systems. Computing in Science & Engineering 2019;21(4):7–21. doi: 10.1109/MCSE.2019.2906593

94. Beltrán Mora D, Castrillo M, Marciani MG, Kinoshita BP, Tenorio-Ku L, Gaya-Àvila A, et al. Autosubmit 4.0.100. Zenodo, 2023. doi: 10.5281/zenodo.10199020

95. Goble C, Cohen-Boulakia S, Soiland-Reyes S, Garijo D, Gil Y, Crusoe MR, et al. FAIR Computational Workflows. Data Intelligence 2020;2(1-2):108–121. doi: 10.1162/dint_a_00033

96. Kinoshita BP. RO-Crate created using Autosubmit version 4.0.100 workflow running kinow/auto-mhm-test-domains. Zenodo, 2023. doi: 10.5281/zenodo.8144612

97. Samaniego L, Kumar R, Attinger S. ~~Implications of distributed hydrologic model parameterization on water fluxes at multiple scales and locations~~Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. Water Resources Research~~2013;49(1):360–379~~, 2010;46(5). doi: 10.1029/~~2012WR012195~~2008WR007327

98. Kumar R, Samaniego L, Attinger S. ~~Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale~~Implications of distributed hydrologic model parameterization on water fluxes at multiple scales and locations. Water Resources Research ~~, 2010;46(5)~~2013;49(1):360–379. doi: 10.1029/~~2008WR007327~~2012WR012195

99. Leo S. Run of digital pathology tissue/tumor prediction workflow. Zenodo, 2023. doi: 10.5281/zenodo.7774351

100. The Galaxy Community. Galaxy. Version 23.1 Software Heritage Archive, 2023. https://identifiers.org/swh:1:rel:33ce0ce4f6e3d77d5c0af8cff24b2f68ba8d57e9

101. Colonnelli I. StreamFlow run of digital pathology tissue/tumor prediction workflow. Zenodo, 2023. doi: 10.5281/zenodo.7911906

102. ~~Fernández JM, Rodríguez-Navas L, Muñoz-Cívico A, Iborra P, Lea D. WfExS-backend. Version 0.10.1. Zenodo, 2023. doi: 10.5281/zenodo.10068956~~

103. Del Rio M, Lianas L, Aspegren O, Busonera G, Versaci F, Zelic R, et al. AI Support for Accelerating Histopathological Slide Examinations of Prostate Cancer in Clinical Studies. Image Analysis and Processing. ICIAP 2022 Workshops. ICIAP 2022. Lecture Notes in Computer Science 2022;13373. doi: 10.1007/978-3-031-13321-3_48

104. CRS4 Digital Pathology Platform [cited 2024 May 27]. https://github.com/crs4/DigitalPathologyPlatform

105. RO-Crate profiles [cited 2024 July 1]. https://www.researchobject.org/ro-crate/profiles.html#ro-crate-profiles

106. MIRAX format [cited 2024 May 27]. https://openslide.org/formats/mirax/

107. Common Provenance Model RO-Crate profile [cited 2024 May 27]. https://w3id.org/cpm/ro-crate

108. Wittner R, Mascia C, Gallo M, Frexia F, Müller H, Plass M, et al. Lightweight Distributed Provenance Model for Complex Real–world Environments. Scientific Data 2022;9:503. doi: 10.1038/s41597-022-01537-6

109. Wittner R, Holub P, Mascia C, Frexia F, Müller H, Plass M. et al. Towards a Common Standard for Data and Specimen Provenance in Life Sciences. Learning Health Systems 2023;e10365. doi: 10.1002/lrh2.10365

110. Wittner R, Gallo M, Leo S, Soiland-Reyes S. Packing provenance using CPM RO-Crate profile. Version 1.1. Zenodo, 2023. doi: 10.5281/zenodo.8095888

111. Wittner R, Soiland-Reyes S, Leo S, Meurisse M, Hermjakob H. BY-COVID D4.3 Provenance model for infectious diseases. Zenodo, 2024 doi: 10.5281/zenodo.10927253

112. The W3C SPARQL Working Group. SPARQL 1.1 Overview. W3C Recommendation 21 March 2013 [cited 2024 May 27]. https://www.w3.org/TR/sparql11-overview/

113. Ferreira da Silva R, Badia RM, Bala V, Bard D, Bremer PT, Buckley I, et al. Workflows Community Summit 2022: A Roadmap Revolution. arXiv:2304.00019, 2023. doi: 10.48550/arXiv.2304.00019

114. Rehm HL, Page AJH, Smith L, Adams JB, Alterovitz G, Babb LJ, et al. GA4GH: International policies and standards for data sharing across genomic research and healthcare. Cell Genomics 2021;1(2):100029. doi: 10.1016/j.xgen.2021.100029

115. de Wit R. 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. Zenodo, 2022. doi: 10.5281/zenodo.7113250

116. de Wit R, Crusoe MR. Analysis of runcrate. Zenodo, ~~2023. doi:~~ 2024. doi: 10.5281/zenodo~~.10251812~~.12689424

117. Leo S, Crusoe MR, Rodríguez-Navas L, Sirvent R, Kanitz A, De Geest P, et al. Recording provenance of workflow runs with RO-Crate (RO-Crate and mapping). Zenodo, 2023. doi: 10.5281/zenodo.10368990

118. Leo S, Crusoe MR, Rodríguez-Navas L, Sirvent R, Kanitz A, De Geest P, et al. Recording provenance of workflow runs with RO-Crate (RO-Crate and mapping). HTML preview [cited 2024 May 27]. https://w3id.org/ro/doi/10.5281/zenodo.10368989

119. Soiland-Reyes S, Wheater S. Five Safes RO-Crate profile. Version 0.4. TRE-FX Candidate Recommendation, 2023 [cited 2023 Dec 11]. https://w3id.org/5s-crate/0.4

120. Giles T, Soiland-Reyes S, Couldridge J, Wheater S, Thomson B, Beggs J, et al. TRE-FX: Delivering a federated network of trusted research environments to enable safe data analytics. Zenodo, 2023. doi: 10.5281/zenodo.10055354

121. Desai T, Ritchie F, Welpton R. Five Safes: designing data access for research. Economics Working Paper Series, 2016;1601. https://econpapers.repec.org/RePEc:uwe:wpaper:20161601

122. Snowley K, Edwards L, Crosby B, Tatlow H. Integrating Our Community. Year 1. Health Data Research UK, 2023 (report) [cited 2023 Dec 11]. https://www.hdruk.ac.uk/wp-content/uploads/2023/10/Integrating-Our-Community_v1-Oct-2023-compressed.pdf

123. EOSC-ENTRUST: Creating a European network of TRUSTed research environments [cited 2024 May 27]. https://eosc-entrust.eu/

124. Mazumder R, Simonyan V (eds). IEEE P2791 BioCompute Working Group (BCOWG). IEEE Standard for Bioinformatics Analyses Generated by High-Throughput Sequencing (HTS) to Facilitate Communication. IEEE Std 2791-2020, 2020. doi: 10.1109/IEEESTD.2020.9094416

125. Alterovitz G, Dean D, Goble C, Crusoe MR, Soiland-Reyes S, Bell A. Enabling Precision Medicine via standard communication of NGS provenance, analysis, and results. PLOS Biology 2018;16(12):e3000099. doi: 10.1371/journal.pbio.3000099

126. Stian Soiland-Reyes. Packaging BioCompute Objects using RO-Crate [cited 2024 May 27]. https://biocompute-objects.github.io/bco-ro-crate/

127. Soiland-Reyes S. Describing and packaging workflows using RO-Crate and BioCompute Objects. Zenodo, 2021. doi: 10.5281/zenodo.4633732

128. ~~Leo S. Workflow Run RO-Crate Introduction. Galaxy Training Materials, 2023 cited 2023 Dec 11.~~

Table 2. Summarised results of our qualitative analysis of Provenance Run Crates generated with runcrate.

~~Type~~CWL (non-RDF)	~~Subtype~~Type	~~Name~~Subtype	~~CWL~~Name	CWLProv RDF	~~RO-Crate~~
●	T1	SC1	Workflow design	~~●~~·	
○	~~…~~	SC2	Entity annotations	·	
·	~~…~~	SC3	Workflow execution ann.	·	
○	T2	D1	Data identification	~~○~~·	
○		D2	File characteristics	○	
○		D3	Data access	~~○~~·	
●		D4	Parameter mapping	●	
●	T3	SW1	Software identification	~~○~~·	
●		SW2	Software documentation	·	
●		SW3	Software access	·	
●	T4	WF1	Workflow software	~~●~~○	
●		WF2	Workflow parameters	~~●~~○	
●		WF3	Workflow requirements	~~●~~·	
·	T5	ENV1	Software environment	·	
·	~~·~~	ENV2	Hardware environment	·	
○		ENV3	Container image	○	
·	T6	EX1	Execution timestamps	~~·~~●	
·		EX2	Consumed resources	·	
·	~~·~~	EX3	Workflow engine	~~·~~○	
·		EX4	Human agent	~~·~~●	

~~We compared RO-Crates with the CWLProv ROs from which they were generated. The analysis was based on a provenance taxonomy reflecting~~ We converted CWLProv (v0.6.0) ROs to WRROC with runcrate 0.5.0. The table compares the degree to which the data subtypes of the provenance data taxonomy (identified by the triple (`Type`, `Subtype`, `Name`)) are preserved by the CWLProv RDF and the WRROC RDF models; the taxonomy is defined in previous work [115], where relevant provenance metadata are identified based on realistic use cases for ROs associated with a real-life bioinformatics workflow ~~[115].~~. For completeness, the *CWL (non-RDF)* column also reports the non-RDF representation of provenance metadata in CWL-specific documents~~are~~: `packed.cwl` (the workflow) ~~,~~and `primary-job.json` (the ~~inputs file), and `primary-output.json` (the outputs file).~~ input parameter file). Since `packed.cwl` ~~is~~ and `primary-job.json` are also included in RO-Crate, we only considered how the metadata was represented in `ro-crate-metadata.json`.
~~For completeness we also show the theoretical capability of the Provenance Run Crate profile (WRROC column) assuming all its MUST/SHOULD requirements are complete. The categories in the first three columns are explained in [115].~~ **Legend:** ● fully represented ○ partially represented · missing or unstructured representation ~~… optional (e.g. schema.org attribute)~~

Table 3. Mapping from Workflow Run RO-Crate to equivalent W3C PROV concepts using SKOS [43]. For instance, ~~*CreateAction*~~ *s:CreateAction* has **broader** match ~~PROV's *Activity*~~*prov:Activity*, meaning that ~~*Activity*~~ *prov:Activity* is more general. Prefix *prov:* `https://www.w3.org/ns/prov#`.

RO-Crate	Relationship	W3C PROV-O
~~*Action*~~ *s:Action* (superclass of ~~*CreateAction, OrganizeActions*~~*s:CreateAction*, *s:OrganizeAction*)	Has close match (~~schema~~Schema.org Actions may also be potential actions in the future)	*prov:Activity*
~~*CreateAction, OrganizeAction*~~ *s:CreateAction*, *s:OrganizeAction*	Has broader match	*prov:Activity*
~~*Person*~~ *s:Person*	Has exact match	*prov:Person*
~~*Organization*~~ *s:Organization*	Has exact match	~~*OrganizeAction*~~*prov:Organization*
~~*SoftwareApplication*~~ *s:SoftwareApplication*	Has related match	*prov:SoftwareAgent*
~~*ComputationalWorkflow, SoftwareApplication, HowTo*~~ *bioschemas:ComputationalWorkflow*, *s:SoftwareApplication*, *s:HowTo*	Has broader match	*prov:Plan*, *prov:Entity*
~~*File, Dataset, PropertyValue*~~ *s:MediaObject*, *s:Dataset*, *s:PropertyValue*	Has broader match	*prov:Entity*
~~*startTime* on *CreateAction*~~ *s:startTime* on *s:CreateAction*	Has close match	*prov:startedAtTime*
~~*endTime* on *CreateAction*~~ *s:endTime* on *s:CreateAction*	Has close match	*prov:endedAtTime*
~~*agent* on *CreateAction*~~ *s:agent* on *s:CreateAction*	Has related match	*prov:wasStartedBy*, *prov:wasEndedBy*
~~*agent* and *instrument* on *CreateAction*~~ *s:agent* and *s:instrument* on *s:CreateAction*	Has broader match	*prov:wasAssociatedWith*
~~*instrument* on *CreateAction*~~ *s:instrument* on *s:CreateAction*	Has related match (Complex mapping: an instrument implies a qualified association with the agent, linked to a plan)	*prov:hadPlan* on *prov:Association*
~~*object* on *CreateAction*~~ *s:object* on *s:CreateAction*	Has exact match	*prov:used*
~~*result* on CreateAction~~ *s:result* on *s:CreateAction*	Has close match	inverse *prov:wasGeneratedBy*