CWLProv – Interoperable Retrospective Provenance capture and its challenges

Farah Z. Khan[1,3], Stian Soiland-Reyes[2, 3], Michael R. Crusoe[3], Andrew Lonie[1], Richard O. Sinnot[1]
[1] The University of Melbourne, Australia, [2] The University of Manchester, UK
[3] Common Workflow Language Project
Email: farah.khan@unimelb.edu.au, soiland-reyes@manchester.ac.uk

Abstract. The automation of data analysis in the form of scientific workflows is a widely adopted practice in many fields of research nowadays. Computationally driven data-intensive experiments using workflows enable Automation, Scaling, Adaption and Provenance support (ASAP). However, there are still several challenges associated with the effective sharing, publication, understandability and reproducibility of such workflows due to the incomplete capture of provenance and the dependence on particular technical (software) platforms. This paper presents CWLProv, an approach for retrospective provenance capture utilizing open source community-driven standards involving application and customization of workflow-centric Research Objects (ROs). The ROs are produced as an output of a workflow enactment defined in the Common Workflow Language (CWL) using the CWL reference implementation and its data structures. The approach aggregates and annotates all the resources involved in the scientific investigation including inputs, outputs, workflow specification, command line tool specifications and input parameter settings. The resources are linked within the RO to enable re-enactment of an analysis without depending on external resources. The workflow provenance profile is represented in W3C recommended standard PROV-N and PROV-JSON format to capture retrospective provenance of the workflow enactment. The workflow-centric RO produced as an output of a CWL workflow enactment is expected to be interoperable, reusable, shareable and portable across different platforms. This paper describes the need and motivation for CWLProv and the lessons learned in applying it for ROs using CWL in the bioinformatics domain.

Keywords: Provenance, Common Workflow Language, Research Object, Retrospective Provenance.

1 Introduction

The transparent and comprehensive sharing of experimental designs is critical to establish trust and ensure authenticity, quality and reproducibility of any research result. With data growing exponentially in different domains [1], the practice to perform computational analyses of generated data using workflows has overtaken many traditional research methods using ad-hoc scripts in past few decades [2]. Scientific workflow design and management has become an essential part of many computationally driven data-intensive analyses enabling Automation, Scaling, Adaptation, and Provenance support (ASAP) [3]. A number of studies have advocated for complete provenance tracking of scientific workflows to ensure transparency, reproducibility, analytical validity, quality assurance and attribution of (published) research results [4]. Provenance information for workflows is divided into: *Retrospective Provenance*; *Prospective Provenance* and *Workflow Evolution*. *Retrospective provenance* refers to the detailed record of the implementation of a computational task including details of every

executed process together with comprehensive information about the execution environment used to derive a specific data product. *Prospective provenance* refers to the 'recipes' used to enact a computational task, e.g. the workflow specification [5]. *Workflow Evolution* refers to tracking of any alteration in the existing workflow resulting in another version of the workflow that may produce either the same or different resultant data artefacts. The focus of this study is to demonstrate the application and customization of Research Objects (ROs) [6] produced as output of workflow enactment using the reference implementation of the Common Workflow Language (CWL) [7] to record retrospective provenance. The concept of workflow-centric ROs has been previously considered in [8–10] where it was used for structuring the analysis methods and aggregating the digital resources utilized in a given analysis. The generated ROs in these studies typically aggregated data objects, example inputs, workflow specifications, attribution details, details about the execution environment, abstract workflow sketch and various other elements.

Many studies have empirically investigated the role of automated computational methods in form of workflows and published best practice recommendations to support workflow preservation, validity, understandability and re-use. We summarise such recommendations below from the literature to develop a consolidated understanding of the framework of a workflow-centric RO.

Table 1. Best practice recommendations for workflow publishing and sharing

R1: Workflows should be treated as first class data objects [10];
R2: Workflow specification alone is insufficient to ensure reusability of scientific experiments [11]. Complete provenance capture of workflow enactment should be published along with the workflow specification. This can also help to avoid workflow decay [8] [12].
R3: A structured description of the experimental steps carried out in a workflow using a "system-neutral" language can ensure well-documented and well described workflows for enhanced understandability of methods [8].
R4: Availability of the underlying associated software with each step of a given workflow is crucial. More recently container technologies such as Docker, OpenVZ or LXC containers can be exploited to package the environment and configuration together [13].
R5: While publishing digital scholarly objects, open licensing should be adopted as a practise to allow sharing and reproducing of published analyses [14, 15].
R6: The description of the underlying software is not enough for reproducing an analysis; instead workflow specifications and configuration should also be published [15].
R7: Intermediate data products should be captured if feasible to facilitate debugging and error handling and thorough examination of the published workflows and associated (intermediate) results [15, 16].

Keeping in view these recommendations, this paper demonstrates how generation of workflow-centric ROs as an artefact of CWL workflow enactment can be used to facilitate the validation of research findings and assist the reuse, sharing and better insights

of results. We argue that the combination of these two standards offers an important step towards achieving comprehensive and executable workflow-centric objects to ultimately improve the understanding and repeatability of scientific investigations. The capture of retrospective provenance as a "*Workflow provenance profile*" leverages well-established community driven ontologies such as PROV [17] and RO vocabularies [18]. The automatically generated RO aggregates all resources utilized in a workflow run such that the framework of the generated RO makes the resources accessible and comprehensible (as detailed in Section 3). The workflow specification and input parameter files are reconfigured to make the workflow packaged in a given RO executable without depending on any local resource configuration dependencies.

The concept of workflow-centric ROs has been implemented previously but compared to previous efforts, we demonstrate how utilizing the principles of interoperability supported by CWL we are able to avoid platform dependent solutions. With a plethora of heterogeneous workflow definition approaches [19], the widely used workflow definition standard CWL is supported by various working groups and organizations implementing scientific workflows in their data analyses. It is defining a standard mechanism for portability, interoperability and reproducibility of analyses between platforms that implement the CWL standards. This provides an interoperable bridge overcoming existing gaps due to lack of consensus and system heterogeneity. In addition, CWL supports the Docker [20] software container format and CWL implementations use Docker, Singularity [21], udocker [22] and other Docker compatible container runtime engines. resulting in independence from any underlying analysis environment and the availability of particular tool versions used in a given analysis leading to preservation of the software environment.

In this paper we discuss the standards utilized for this work in section 2 followed by implementation details and demonstration of the implemented module for an example CWL workflow in section 3. Section 4 identifies limitations and challenges of interoperable provenance capture and details current and future work. Section 5 considers related work in the area of workflow-centric ROs and especially those capturing retrospective provenance and finally Section 6 concludes the work.

2 Applied Standards and Vocabularies

In this work we follow a recommendation "*Reuse vocabularies, preferably standardized ones*" from best practices associated with data sharing, representation and publication on web to achieve consensus and interoperability of workflow-based analyses. We have integrated mature community-driven standards to accomplish aggregation of all necessary artefacts supporting retrospective provenance associated with workflow enactment. This section discusses the standards and vocabularies applied to achieve an interoperable solution for workflow and provenance publication and sharing.

Common Workflow Language (CWL) aims to provide extensible, open source standards supporting interoperable, portable and reproducible workflow-based research. The need for portable and interoperable standards to promote collaborative research is needed more than ever with the rapidly growing list of workflow management systems and workflow definition approaches. CWL provides declarative constructs for

workflow and command line tool definition and make minimal assumptions about base software dependencies, configuration settings, software versions, parameter settings or the execution environment more generally [13]. It supports comprehensive recording and of information during workflow design and execution, which can subsequently be structured and published alongside any resultant analysis. CWL is a community driven effort widely adopted by workflow design and execution platforms supports interoperability across these diverse platforms. Examples of current adopters include workflow-centric research efforts such as Toil, Arvados, Rabix [23] and Bcbio [24] with in progress implementations in Galaxy, Apache Taverna, and REANA. CWL's object model supports the efficient capture of essential provenance information.

Research Object (RO) An exemplary RO encapsulates all the digital artefacts associated with a given computational analysis. The aggregated resources include but are not limited to: input and output data for experiment results validation; computational methods such as command line tools and workflow specifications to facilitate workflow reruns; attribution details for user authentication; retrospective as well as prospective provenance for better understanding of workflow requirements; and machine-readable annotations regarding the included artefacts and the relation between them. Consequently, the goal is to make any published scientific investigation and the produced artefacts "*interoperable, reusable, citable, shareable and portable*". The three main principles of the RO approach are 'Identity', 'Aggregation', and 'Annotation'. Together they look to enable accessibility of tightly-coupled, interrelated and well-understood aggregated resources involved in a computational analysis as an identifiable object, e.g. using unique identifiers such as DOIs and ORCIDs. The RO approach is well aligned with the idea of interoperable and platform-independent solutions for provenance capture of workflows. While ROs can be serialized in different ways, in this work we have reused the *BDBag* approach based on *BagIt*, which has been shown to support large-scale workflow data [25] as well as being compatible with data archiving efforts in NIH Data Commons, Library of Congress and Research Data Alliance. The specialized workflow-centric RO in this study encompasses the above-mentioned components annotated with various targeted tools and PROV-based "*Workflow provenance profile*" capturing the detailed retrospective provenance of the CWL workflow enactment.

PROV Data Model (PROV-DM) The World Wide Web Consortium (W3C) developed PROV based on the recommendations of the Provenance Incubator Group. It refers to a suite of specifications introduced to support the unified/interoperable representation and publication of provenance on the Web. The underlying conceptual model, PROV Data Model (PROV-DM) provides a domain-agonistic model designed to capture fundamental features of provenance with support for extensions to integrate domain-specific information. We have utilised mainly two serializations of PROV for this study, PROV-Notation (PROV-N) [26] and PROV-JSON [27]. PROV-N is designed to achieve serialization of the PROV-DM instances by formally representing the information using simplified technology-independent syntax to improve readability. PROV-JSON is a lightweight interoperable representation of PROV assertions using JavaScript constructs and data types. The key design and implementation principles of these two serializations of PROV are in compliance with the goals of this study, hence are a natural choice to support the design of an adaptable provenance profile.

3 CWLProv

We present an implementation (CWLProv) integrating the underlying principles of the applied standards to generate a re-enactable and interoperable workflow object.

3.1 Framework of Workflow Research Object

After analysing the set of recommendations, we modelled the structure of the RO to identify the data and metadata required accordingly. We systematized the aggregated resources into the following collections for better understanding and accessibility (**Fig. 1**). The structure of the RO follows the BDBag approach with checksums of the payload data/ directory listed in the BagIt manifest-sha1.txt and the RO manifest as JSON-LD in metadata/manifest.json. The remaining directories constitute the metadata of how the workflow results were created.

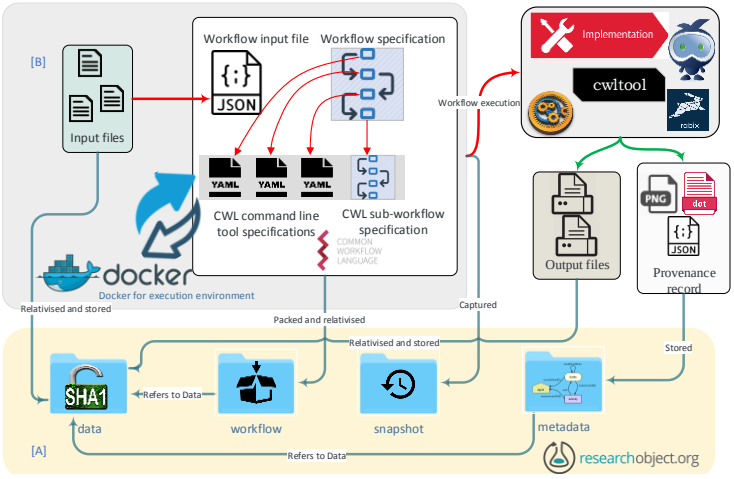

Fig. 1. Schematic representation of aggregation and links between components of an RO. Each CWL tool specification can optionally interact with Docker to satisfy software dependencies. The RO layer (yellow, [A]) shows the structure of the RO including its contents and interaction with different components in the RO and the CWL layer (grey, [B]).

data/ is the payload collection of all input and output files used in the workflow enactment, named according to their SHA-1 hash checksum rather than derived from their multiple occurrences during workflow execution. This use of content-addressable storage [28] simplifies identifier generation for data values in the workflow engine.

workflow/ contains copies of the workflow specification file and a input object in JSON. These do not match exactly the executed files; the absolute paths in the input job file are replaced with relativized content-addressed paths in "data/". In addition, the "--pack" method of cwltool is facilitated to aggregate the CWL description and any referenced external descriptions (such as sub workflows or external command line tool

descriptions) into a single executable file. The workflow files are thus rewritten to be re-runnable without depending on files outside this RO.

snapshot/ comprises copies of the workflow and tool specifications files "*as-is*". We recommend using these resources just for validity of results and for understanding the workflow enactment, since these files might contain absolute paths or be host-specific; hence cannot necessarily be re-enacted elsewhere.

metadata/ The workflow provenance profile for the workflow execution associated with the RO requires rich metadata. PROV-N is used to design the example template which is later translated into a PROV-JSON and PROV-N file using the Prov Python library which supports serialization the W3C Provenance Data Model to JSON, PROV-N and other formats. In addition, *wfdesc* is utilized to describe the abstract representation of the workflow and its steps. *wfprov* is applied to capture provenance aspects that are hard to present otherwise when using the PROV serialization alone.

3.2 Workflow Enactment Provenance Profile

Building on the core elements of a RO and considering the structure, we propose an example provenance profile represented using the PRON-N ontology. It refers to a sample two-step workflow where output of the first step is used as input to the second step. Terms below are used in the retrospective provenance profile associated with the CWL workflow enactment.

wfdesc "*wfdesc:Workflow*" for the workflow specification and "*wfdesc:hasSubProcess*" to relate individual steps to the workflow is added. Each step is described by "*wfdesc:Process*". This gives the prospective view of the workflow specification without the requirement for executing it, hence is associated with entities.

wfprov is used for the activities and entities in the provenance profile. "*wfprov:WorkflowRun*" and "*wfprov:ProcessRun*" represents the workflow and individual command line tool execution respectively. The data items are also described by "*wfprov:Artifact*".

Entity refers to any data artefact, input configuration files, workflow and command line tool specification. Each data artefact is identified using SHA-1 hash value used to store files in *"data"*.

Activity: A workflow run and command line tool invocations are classified as activities identified by a Universal Unique Identifier and labelled with the absolute name of the step given in the workflow file to improve the readability.

Agent: "*SoftwareAgent*" represent the engine used to execute the Workflow Plan, identified by a UUID.

wasAssociatedWith: relates activities such as WorkflowRun and ProcessRun to the *SoftwareAgent*.

wasStartedBy records an activity's identifier, starter and time of commencement. In case of a *WorkflowRun*, the starter is not depicted as the workflow enactment is not triggered by another activity whereas in case of a *ProcessRun*, the starter is the corresponding *WorkflowRun*.

Used is used by *WorkflowRun* and *ProcessRun*. The same entity representing a data artefact can be used by *WorkflowRun* and *ProcessRun* at different instances of time.

wasGeneratedBy is used to associate all output data artefacts, represented as entities with the respective activity that generates them.

wasEndedBy represents the relationship between *ProcessRun* and *WorkflowRun* and the time when the *ProcessRun* is ended by the *WorkflowRun*.

3.3 Implementation of CWLProv

To demonstrate the practical realisation of the above described workflow-centric RO and provenance profile, a feature complete implementation *cwltoo*l was adopted. It incorporates extensive validation of the CWL files as well as offering a comprehensive set of test cases to validate new modules introduced for extension to the existing implementation. The existing classes and methods of the implementation were also utilized to achieve various tasks such as packaging of the workflow and all the tool specifications. CWLProv is an optional module which when invoked as "*cwltool --provenance `RO-name` workflow.cwl job.json",* will automatically generate an RO with the given name without requiring any additional information from the user. The information about the input files is extracted from the JSON job object. Each input file is assigned a SHA-1 hash value and placed in "*data*", making it content-addressable to avoid local references (**Fig. 2**). In the next step the workflow and command line tool specifications are aggregated in one file to create an executable workflow. Moreover, the input object is transformed into a job object with the references to artefacts in the RO-name/data by relativising the paths present in the input object. The *cwltool* control flow indicates the points when the execution of the workflow and command line tools involved in a workflow enactment start, end and how the output is reported back. CWLProv utilises this information and collects the artifacts to be captured in the RO. It continually updates the provenance profile throughout this process.

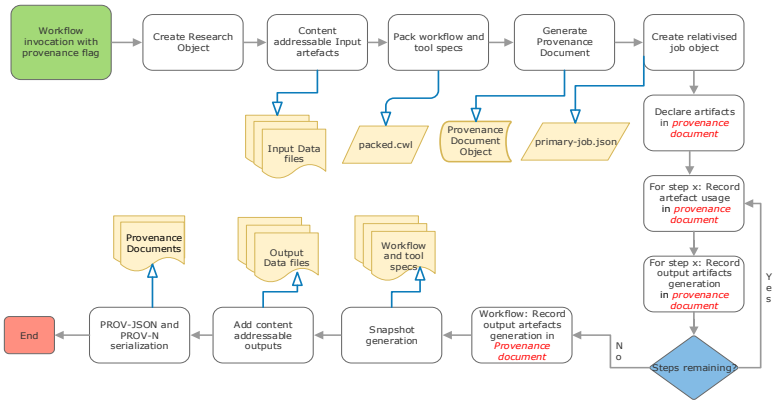

Fig. 2. High level process flow representation of retrospective provenance capture

When the execution of the workflow begins, CWLProv generates a document (using the *prov* library) which is updated with the default namespaces and the workflow run "activity". In addition, a UUID for the engine is also generated to record the *SoftwareAgent* for every workflow enactment. For each step, a UUID is generated as an identifier for the "activity" to be included in the provenance profile. For each step level

activity, the start time and association with the workflow activity is created and stored as part of the overall provenance. After completing the execution of an individual step, the outputs are assigned to the outputs of the workflow step and the provenance profile records the generation of outputs at the step level. Once all steps complete, the workflow outputs are collected and the generation of these outputs at the workflow level are recorded in the provenance profile. Moreover, using the checksum generated by *cwltool*, the content-addressable copies are saved in "data". The provenance profile refers to these files using the same checksum such that they are traceable for further analysis if required. The workflow specification, command line tool specifications and JSON job file is archived in the "*snapshot*" to preserve the actual workflow history.

3.4 Case Study Demonstration

The CWL workflow used to demonstrate working of CWLProv accepts protein and respective nucleotide sequence files as input and compute ratio of number of non-synonymous substitutions per non-synonymous sites to number of synonymous substitution per synonymous sites (*dN/dS*). *dN/dS* ratio is frequently used in evolutionary studies as a measure to quantify selective pressure on a protein coding region using codon as unit of evolution [29]. The workflow comprises of two array type input ports for each input and one step invoking a nested sub workflow (**Fig. 3**).

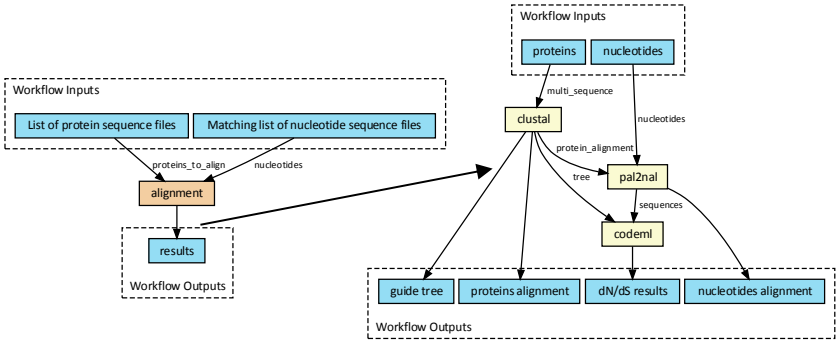

Fig. 3. Representation of the workflows where alignment step enacts the nested sub workflow comprising of three steps (images: CWL viewer)

The "scatter" operation in CWL when applied to one or more input parameters of a workflow step or a sub workflow, supports parallel execution of the associated process. Parallelism is also available without "scatter" when separate processes have all their inputs ready. If enough compute resources are available these jobs will be enacted concurrently, else queued for execution. Compute intensive steps of a workflow can benefit from scatter feature for parallel execution and reduce overall run time. Both input parameters in this workflow are using "scatter" feature specifying that the nested workflow should be enacted separately for each protein and its respective nucleotide sequence file. In case of more than one input parameter using scatter, a "scatterMethod" is also required to understand decomposition of inputs into set of independent jobs. This

workflow is using “dot product” as scatter method which implies that input lists are aligned in such a way that one element is extracted from each list for each enactment of the nested workflow. We enacted this workflow using three example protein sequence and respective nucleotide sequence files from example data. For each set, the sub workflow was enacted as an isolated job resulting into three nested executions of the workflow. The generated workflow-run RO available on GitHub, is tested within the “*workflow*” directory as “*cwltool packed.cwl primary-job.json*” for successful re-enactment using the relativised data and resources in the RO.

4 Challenges and Limitations

Our case study is a concise example of a conventional workflow including a nested sub workflow for modular structure and the use of the CWL scatter feature for parallel execution of independent tasks using the same tool. The sub workflow currently is treated as a black box and not catered as a separate “WorkflowRun” in the provenance profile. As a result, the generated provenance profile is a flattened view of the relationship between the activities and entities of the outer and nested workflow without sufficient distinction of the data flow from sub workflow to the main workflow (**Fig. 4**). Content-addressable data will deal with the artefact name mismatch when mapping data between the nested and main workflow. For content-addressing we reused cwltool’s SHA-1 hash values, however this should be replaced with a stronger hash like SHA-256 as SHA-1 has been proven to have predictable collisions [30]. The choice of identifier scheme for data values is not currently fixed in CWLProv, as different CWL implementations may have existing hash or data identifier mechanisms that are natural to reuse. However, this means implementations could generate different identifiers for the same value, making it harder to cross-reference equal data values in multiple ROs. In addition, currently we are using UUIDs as identifiers for activities, but server-hosted CWL implementations might have existing URIs that could be reused for activity identifiers.

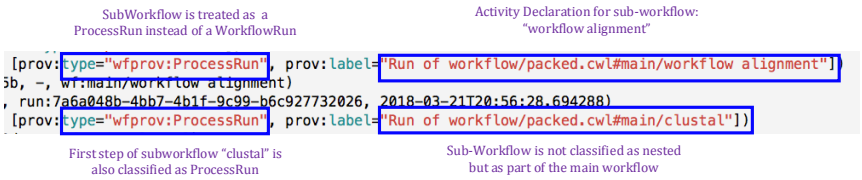

Fig. 4. showing a nested workflow treated as a ProcessRun only in Provenance Profile

The provenance capture and RO customization can be classified into four levels to realise a hierarchical and modular solution towards interoperable domain specific provenance capture (**Fig. 5)**. Level 0 refers to the lowest level of provenance in which the RO only contains the actual files used in a workflow enactment. In this case it can only be used for result interpretation and debugging in case of failed executions. Level 1 builds upon this to add retrospective provenance profiles, with content-addressable data and an executable workflow definition to ensure reproducibility of the analysis. These

levels are complete whereas level 2 and 3 are a work in progress. At level 2, we aim to handle nested workflows as a WorkflowRun and achieve reproducibility of main as well as nested workflow such that both specifications are re-runnable independently. In addition, we are working on generating multiple provenance profiles for main and nested workflows to deliver different views for each sub workflow enactment. Furthermore, we are working on adding more PROV constructs such as *prov:alternateOf* and *prov:specializationOf* to include clear references within the RO. For version-controlled workflows, adopting a permalink URI scheme based on git for identifying the workflows can help achieve persistence of the software and mitigate the ambiguity when referring to the data artefacts. In addition, we are planning to incorporate RFC6920 [31] *ni* URIs for global data identifiers, as well as the proposed *arcp* URI scheme for identifying files contained within the RO.

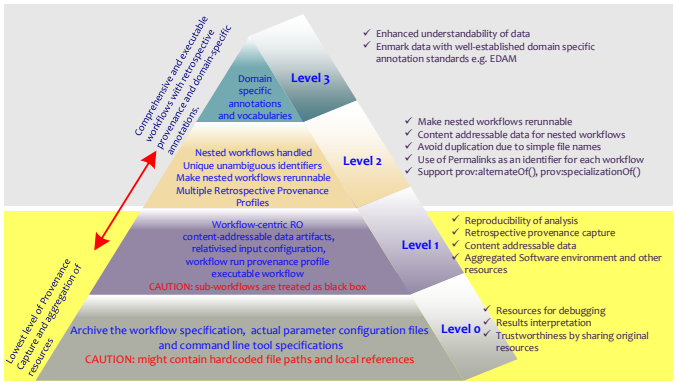

Fig. 5. Depiction of the levels of provenance and resource aggregation.

CWL supports incorporation of domain-specific ontologies to describe data artefacts. On level 3 we aim to extract such information from the CWL workflow specification and utilize it to annotate the entities in the provenance profiles for understandability of the data artefacts used in a given analysis. CWL also supports declaration of attribution details such as author name, institute and email address. This information can be represented using *prov:agent()* in the workflow prospective provenance profile as well as with *pav:authoredBy* in the RO manifest to propagate attribution details.

5 Related Work

The research related to our study can be categorized into two classes: previous studies implementing workflow centric ROs and efforts focusing on documenting retrospective provenance associated with workflow enactments.

Application of Workflow-centric ROs. Belhajjame et al. [8] proposed the application of ROs to develop workflow-centric ROs containing data and metadata that supports the understandability of the analysis methods. They explored five essential requirements to workflow preservation and identified data and metadata that can be stored to satisfy the said requirements. They proposed extension to existing ontologies and

developed four new ontologies to represent workflow specific information. However, the scope of the proposed model at that time was not interoperability in terms of execution of the aggregated workflows as it was demonstrated for a Taverna workflow using myExperiment which makes it quite platform-dependent. Gomez-Perez et al. proposed extensions to the RO model to equip a workflow-centric RO with information catering for the specific needs of the Earth Science community, resulting in enhanced findability and reusability for experts [32]. They demonstrated that the principles of the RO in general support extensions to generate aggregated resources leveraging domain specific knowledge. Hettne et al. used genomic workflows as a case study to demonstrate the utilization of ROs to capture methods and data supporting querying and useful extraction of information about the scientific investigation under observation [9]. The solution is tightly coupled with the Taverna Workflow Management System and hence if shared, would not be interoperable or rerunnable outside of the Taverna environment.

Tracking Retrospective Provenance. Plethora of studies are working in various dimensions to capture retrospective provenance by using established tools and standards or proposing a new solution to the problem. We only consider studies utilizing the Provenance standards (PROV specifications) in any capacity. Prabhune et al. adopt the principles of ProvONE and introduce a system comprised of three components including a 'Provenance Manager', responsible for handling the provenance information generated at any stage of workflow life cycle. The Prov2ONE module as part of provenance manager, implements an algorithm designed to automate the construction of provenance graphs involving capture of prospective and retrospective provenance. Michaelides et al. support portability and reproducibility of a statistical suite [33] by capturing the essential elements from the log of a workflow run, representing them using an intermediate notation and later translate to PROV-N. A Linux-specific provenance approach was proposed by Paquier et al. where they demonstrated retrospective provenance capture at the system level [34]. Another ongoing project UniProv is working to extract information from Unicore middleware and transform it into PROV-O representation to facilitate the back-tracking of an experiment [35]. The toolkit, PROV-man uses the PROV standard to record provenance information that can be created, managed and queried using the programming API and stored in a configurable relational database [36]. Platforms as VisTrials [37] and Taverna have built in retrospective provenance support. Taverna implements an extensive provenance capture system "Taverna Provenance Suite" utilizing both PROV ontologies as well as ROs aggregating the resources used in an analysis. Vistrails is an open source project supporting platform-dependent provenance capture, visualization and querying for extraction of required information about a workflow run. Chirigati et al. provide an overview of PROV terms and how they can be translated from the VisTrials schema and serialized to PROV-XML [38] .

6 Conclusion

This work has been implemented observing best practice recommendations in current literature (**Table 1**), resulting in mapping the recommendations (R1..Rn) from literature

(discussed in section 1) to the design choices (regarding standards) made (defined as Action A1..An):

Table 2. Mapping recommendations to realisations of this study

A1: We have focused on "*workflow-centric*" ROs for storing and sharing workflows similar to any other data artefact.
A2: A provenance profile is generated for recording retrospective provenance of the CWL workflow enactment.
A3: A community-driven widely adopted standard CWL was selected for the CWLProv implementation. This directly supports the documentation and description of the processes using standard constructs.
A4: CWL supports Docker format software containers and CWL implementations use Docker compatible container runtimes such as Docker itself, Singularity, and udocker to solve dependency issues associated with the software analysis environment.
A5: All the standards and implementations including CWLProv are community driven, open source and based on open licensing efforts.
A6: The job file containing the parameters used to run the CWL workflow along with the input data was aggregated in the workflow-centric RO to support reproducibility.
A7: Intermediate data capture is facilitated by CWL in such a way that the outputs when declared at "workflow-level" are captured as workflow outputs that could be used for debugging and further analysis purposes.

We have identified hierarchical levels of provenance and resource aggregation and make the case for reusing existing best practice open-source standards for workflow definition (CWL), for aggregation and annotation of resources (RO), and for provenance representation (PROV-DM, RO ontology). Achieving highest level of provenance and aggregation (**Fig. 5**) will facilitate the validation of scientific findings, support reruns and debugging, offer understanding of the data artefacts and establish trustworthiness. Our approach will mitigate the *workflow decay* and underlying principles of the standards utilized to implement CWLProv will result in a semantically rich executable workflow objects such that any platform supporting CWL and CWLProv will be able to reproduce them. We ultimately aim to achieve a solution which aligns with all four dimensions of FAIR principles [39]. In this study, provenance capture and aggregation of resources at Level 1 has resulted in achieving reusability. This study can further be extended to support provenance capture for other implementations of CWL to demonstrate interoperability. Our approach to level 2 and 3 will result in achieving findability and accessibility dimensions to satisfy the requirements of all four dimensions of FAIR principles.

Acknowledgements: We gratefully acknowledge The University of Melbourne for providing funding support as MIRS and MIFRS scholarship. SSR was funded by European Union contract H2020-EINFRA-2015-1-675728 (BioExcel CoE).

7 References

1. Stephens ZD, Lee SY, Faghri F, et al (2015) Big Data: Astronomical or Genomical? PLoS Biol 13:e1002195
2. Atkinson M, Gesing S, Montagnat J, Taylor I (2017) Scientific workflows: Past, present and future. Future Gener Comput Syst 75:216–227
3. Cuevas-Vicenttín V, Dey S, Köhler S, et al (2012) Scientific Workflows and Provenance: Introduction and Research Opportunities. Datenbank-Spektrum 12:193–203
4. Herschel M, Diestelkämper R, Ben Lahmar H (2017) A survey on provenance: What for? What form? What from? VLDB J 26:881–906
5. Clifford B, Foster I, Voeckler J-S, et al (2008) Tracking provenance in a virtual data grid. Concurr Comput 20:565–575
6. Bechhofer S, Buchan I, De Roure D, et al (2013) Why linked data is not enough for scientists. Future Gener Comput Syst 29:599–611
7. Amstutz P, Crusoe MR, Singh M, et al (2017) common-workflow-language/cwltool
8. Belhajjame K, Zhao J, Garijo D, et al (2015) Using a suite of ontologies for preserving workflow-centric research objects. Web Semantics: Science, Services and Agents on the World Wide Web 32:16–42
9. Hettne KM, Dharuri H, Zhao J, et al (2014) Structuring research methods and data with the research object model: genomics workflows as a case study. J Biomed Semantics 5:41
10. Corcho O, Garijo Verdejo D, Belhajjame K, et al (2012) Workflow-centric research objects: First class citizens in scholarly discourse. In: Proceedings of Workshop on the Semantic Publishing. Facultad de Informática (UPM), p 12
11. Zhao J, Gomez-Perez JM, Belhajjame K, et al (2012) Why workflows break; Understanding and combating decay in Taverna workflows. In: 2012 IEEE 8th International Conference on E-Science. pp 1–9
12. Garijo D, Gil Y, Corcho O (2017) Abstract, link, publish, exploit: An end to end framework for workflow sharing. Future Gener Comput Syst 75:271–283
13. Kanwal S, Khan FZ, Lonie A, Sinnott RO (2017) Investigating reproducibility and tracking provenance – A genomic workflow case study. BMC Bioinformatics 18: . doi: 10.1186/s12859-017-1747-0
14. Stodden V, McNutt M, Bailey DH, et al (2016) Enhancing reproducibility for computational methods. Science 354:1240–1241
15. Garijo D, Kinnings S, Xie L, et al (2013) Quantifying reproducibility in computational biology: the case of the tuberculosis drugome. PLoS One 8:e80278
16. Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten simple rules for reproducible computational research. PLoS Comput Biol 9:e1003285
17. Missier P, Belhajjame K, Cheney J (2013) The W3C PROV Family of Specifications for Modelling Provenance Metadata. In: Proceedings of the 16th International Conference on Extending Database Technology. ACM, New York, NY, USA, pp 773–776
18. Soiland-Reyes S, Bechhofer S, Corcho O, et al (2016) Research Object Ontology. In: Wf4Ever Specification. https://w3id.org/ro/2016-01-28/. Accessed 4 Mar 2018
19. Leipzig J (2017) A review of bioinformatic pipeline frameworks. Brief Bioinform 18:530–536
20. Merkel D (2014) Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux J 2014:
21. Kurtzer GM, Sochat V, Bauer MW (2017) Singularity: Scientific containers for mobility of compute. PLoS One 12:e0177459

22. Gomes J, Campos I, Bagnaschi E, et al (2017) Enabling rootless Linux Containers in multi-user environments: the udocker tool. arXiv [cs.SE]
23. Kaushik G, Ivkovic S, Simonovic J, et al (2017) Rabix: An Open-Source Workflow Executor Supporting Recomputability and Interoperability of Workflow Descriptions. In: Biocomputing 2017, Proceedings of the Pacific Symposium
24. Guimera RV (2012) bcbio-nextgen: Automated, distributed next-gen sequencing pipeline. EMBnet.journal 17:30
25. Chard K, D'Arcy M, Heavner B, et al (2016) I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets. In: 2016 IEEE International Conference on Big Data (Big Data). pp 319–328
26. Moreau L, Misser P, Cheney J, Soiland-Reyes S (2013) PROV-N: The Provenance Notation. W3C
27. Huynh TD, Jewell M, Keshavarz AS, et al (2013) The PROV-JSON Serialization. A JSON Representation for the PROV Data Model. Tech. rep., University of Southampton
28. Services EE (2010) Information Storage and Management: Storing, Managing, and Protecting Digital Information. John Wiley & Sons
29. Muse SV, Gaut BS (1994) A likelihood approach for comparing synonymous and nonsynonymous nucleotide substitution rates, with application to the chloroplast genome. Mol Biol Evol 11:715–724
30. Stevens M, Bursztein E, Karpman P, et al (2017) Announcing the first SHA1 collision. Google Security Blog
31. Farrell S, Dannewitz C, Hallam-Baker P, et al (2013) Naming things with hashes
32. Gomez-Perez JM, Palma R, Garcia-Silva A Towards a Human-Machine Scientific Partnership Based on Semantically Rich Research Objects
33. Michaelides DT, Parker R, Charlton C, et al (2016) Intermediate Notation for Provenance and Workflow Reproducibility. In: Provenance and Annotation of Data and Processes. Springer International Publishing, pp 83–94
34. Pasquier T, Han X, Goldstein M, et al (2017) Practical Whole-system Provenance Capture. In: Proceedings of the 2017 Symposium on Cloud Computing. ACM, New York, NY, USA, pp 405–418
35. Giesler A, Czekala M, Hagemeier B, Grunzke R (2017) UniProv: A Flexible Provenance Tracking System for UNICORE. In: High-Performance Scientific Computing. Springer International Publishing, pp 233–242
36. Benabdelkader A, VanKampen AA, Olabarriaga SD (2015) PROV-man: A PROV-compliant toolkit for provenance management. PeerJ PrePrints
37. Freire J, Silva CT (2012) Making Computations and Publications Reproducible with VisTrails. Computing in Science Engineering 14:18–25
38. Chirigati F, Freire J, Koop D, Silva C (2013) VisTrails Provenance Traces for Benchmarking. In: Proceedings of the Joint EDBT/ICDT 2013 Workshops. ACM, New York, NY, USA, pp 323–324
39. Wilkinson MD, Dumontier M, Aalbersberg IJJ, et al (2016) The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3:160018