Applied Ontology 0 (2020) 1
IOS Press

WePROV: An Ontology for workflow evolution provenance analytics

Anila Sahar Butt * and Peter Fitch
CSIRO Land and Water, Australia.
E-mails: anila.butt@csiro.au, peter.fitch@csiro.au

Abstract. Scientific workflow evolution provenance is essential for tracking, attributing and comparing the scientific research. The data models to capture the provenance, known as provenance models, have been of broad interest and studied in many fields, including the semantic Web. The provenance ontologies proposed by the semantic Web community for scientific workflows describe workflow specifications and their executions; however, they overlook workflow evolution provenance that is imperative for comprehensive provenance analytics in the context of the scientific workflow. In this paper, we present a Workflow Evolution Provenance ontology (WePROV) that defines concepts, features, attributes, and relations to describe a workflow and its evolution process. To this, we discuss the underline provenance model and the design process of the proposed ontology. Further, we define the functional requirements for a workflow evolution ontology and validate the proposed ontology through SPARQL queries for these requirements. We also present WePROV aligned dataset, built using open-source scientific workflows, to demonstrate the real-world applicability of the ontology. Our research shows that an ontology capable of describing the workflows and their evolutionary activities in a generic, machine-readable and interoperable format, enables automatic tracking of workflow changes and traversing between their different versions.

Keywords: Scientific Workflow, Workflow Evolution, Provenance Ontology

1. Introduction

Scientific workflows are the popular mechanism for specifying and automating repetitive experiments (Deelman et al., 2018). A significant aspect of their value lies in their potential to be reused. Once specified and shared, a workflow becomes a useful building block that can be combined or modified for developing new experiments (Belhajjame et al., 2015). These workflows encapsulate a vast amount of knowledge associated with any scientific experiments (Withana et al., 2010). When a workflow is used continuously over an extended period, the research is likely to evolve along different dimensions, which causes the associated workflow to evolve. Maintaining a detailed provenance of this process has many benefits that go beyond documentation and result in reproducibility (Butt et al., 2020). For example, this provenance information can be used to visualise the development of the research to see the path to the current state and what the previous attempts were. Also, some scientists may backtrack the workflow to a prior version and take a new direction. There might be an error in a model or an experiment, and this provenance can trace back the origin of that error. Alternatively, it helps to retrieve data products and results affected by this error.

Various Scientific Workflow Management Systems (SWfMSs) have been proposed and developed to provide an environment for specifying and enacting workflows (e.g., Taverna (Hull et al., 2006), Kepler (Altintas et al., 2004), Triana (Taylor et al., 2007), and YAWL (Van Der Aalst and Ter Hofstede,

*Corresponding author. E-mail: anila.butt@csiro.au.

2005)). State-of-the-art SWfMSs automatically capture workflow provenance in the form of execution traces. There also exist stand-alone approaches to capture and analyse the provenance of scientific workflows (Oliveira et al., 2018; Prabhune et al., 2018). However, most of the systems and approaches do not capture provenance of revising a workflow definition (workflow evolution), rather account the scientist for managing the evolution of workflow through existing means for versioning files, such as filenames and folders, version control systems like git[1], or workflow sharing websites like myExperiment[2]. Only a few SWfMSs (e.g., Kepler (Altintas et al., 2004) and VisTrails (Freire et al., 2006)) proposed solutions to automatically capture the limited amount of workflow evolution provenance. But these solutions rely on proprietary formats that make it difficult to reuse these solutions. To overcome this limitation, there is a need for an ontology to comprehensively represent scientific workflows and their evolution in a generic, machine-readable, and workflow-engine independent format. In this format, they can be annotated, queried, and understood without relying on a language specific to a workflow-engine.

In recent years, the semantic Web community has proposed provenance models to describe scientific workflows and their execution traces in a generic, machine-readable and interoperable format. In this regard, the Wf4Ever (Belhajjame et al., 2015) project addresses challenges related to generically described scientific experiments through the *wfdesc*[3] ontology, and an analysis and management of their execution provenance through the *wfprov*[4] ontology. The Open Provenance Model for Workflows (OPMW) (Garijo and Gil, 2011) presents a framework to publish computational workflows, which includes the specification of the *OPMW*[5] ontology, for the description of workflow traces and their templates. ProvONE (Cuevas-Vicenttín et al., 2016) is another provenance model for scientific workflow, which is compatible with PROV-DM[6] and provides constructs to model workflow specification and workflow execution provenance. All these models can capture, store, and query the provenance of workflows. However, they are unable to specify workflow evolution provenance.

It is critical to understand the workflow evolution handling approach before designing an ontology for scientific workflows provenance. There are three main approaches to handle evolving workflows (Kradolfer and Geppert, 1999). The first approach consists of allowing workflow modifications without retaining the pre-modification state. Each change is applied irreversibly to the workflow, considering possible consequences to the previous execution traces of this workflow. In this approach, tracking or reproducibility is not possible. With the second approach, the state of workflow and its execution traces prior to the modifications are conserved. This approach allows tracking and traversing among versions; however, it requires managing a set of workflow versions. The third approach records only the latest workflow version and its difference (i.e., Δ) from the previous version. This approach can reconstruct the previous versions, on-demand, without having to maintain a version database. This approach is concise and requires substantially less space than the alternative of storing multiple versions of a workflow specification (Roddick, 1995); therefore, we prefer this approach.

Based on above discussion, the main question to answer in this research is: **How can we design an ontology to enable automatic tracking of workflow changes and traversing between its different versions without recording a set of workflow versions?** To answer this question, we propose the Workflow Evolution Provenance ontology (WePROV) whose purpose is to identify and represent the

[1] https://github.com/
[2] https://www.myexperiment.org/workflows
[3] http://wf4ever.github.io/ro/#wfdesc
[4] http://wf4ever.github.io/ro/#wfprov
[5] https://www.opmw.org/model/OPMW/
[6] https://www.w3.org/TR/prov-dm/

fundamental concepts, attributes, and relations present in workflows and their evolution process. The ontology is built in a modular way to focus its reuse. As a part of the ontology design process, we first identify the functional requirements that an ontology must meet to be a comprehensive ontology for the workflow evolution provenance. Secondly, we explore the current ontologies to evaluate their reusability to design WePROV. Finally, we discuss three phases of the workflow lifecycle upon which we define four ontology design patterns to describe the workflow evolution provenance. Also, we present WePROV aligned dataset, built using open-source scientific workflows, to demonstrate the use of the ontology. Further, we validate the proposed ontology for its functional requirements through SPARQL queries. The evaluation shows that an ontology capable of describing the workflows and their evolutionary activities in a generic, machine-readable and interoperable format, enables automatic tracking of workflow changes and traversing between their different versions.

The rest of the paper is organised as follows. In Section 2, we motivate the need for workflow evolution provenance. In Section 3, we present the ontology development process. In Section 4, we discuss the workflow evolution provenance ontology and the underline data model. In Section 5, we evaluate our proposed ontology according to the community-defined best practices. In Section 6, a review of the state-of-the-art is provided. Finally, we conclude our work in Section 7.

2. Motivation

The motives behind workflow evolution provenance stem from having to specify a workflow and reconsider the workflow structure and contents to satisfy the research or performance needs (Koop et al., 2010). We identified three use cases that should be supported by an effective workflow evolution provenance system in close collaboration with SWfMS developers and experts. These experts are leading the development of the Senaps[7] Platform and related projects. One of the expert, a team lead, worked very closely with the clients of the platform to understand the client's provenance needs. The three major use cases that motivated the need for a workflow evolution provenance system are:

2.1. Tracking effects over time

When scientists associate their research with workflows, tracking the evolution of these workflows becomes an approximation to the initial problem of tracking the evolution of their research. Along with the evolution of a workflow, all its components evolve. Scientists should be able to look at the workflow evolution history and reason about how the research progressed at the current level to produce an output. Another critical aspect of tracking the evolution is to track the lineage and failure cause in the experiment. For example, if an algorithm or a workflow is erroneous, the evolution information helps to track the affected component or structure.

2.2. Attribution

When a workflow is specified, attribution information such as who designed the experiment, who owns or created the workflow, who owns the data products can be gathered. This attribution information is useful to track down the issues or to give proper credit to the original owners. Also, while specifying workflows, it is becoming more common to reuse components (i.e., algorithms or models) within a

[7] https://products.csiro.au/senaps/

workflow. Scientists utilise not only the algorithms and implementations developed by others but also the data products generated, including optimally derived model parameter configurations. For example, researchers within Digiscape[8] use the climate data published by the standard bodies to test their models. This reuse of a public corpus reduces effort and increases its acceptance. Tracking this kind of contribution not only provides a way to track contributions but also attribution for proper accreditation to the contributors.

2.3. Comparing Workflow Versions

A given research line might evolve in more than one direction. It is essential to understand the changes in these directions by comparing the difference between two or more versions of the same research. For example, given two outputs of two different versions of a workflow, one can deduce the reason for the difference between the two results by analysing the evolution information.

3. Foundational Material

3.1. Ontology Development Process

The term ontology originates in philosophy. In computer science, there are several definitions of what an ontology is? We adopt the definition found in (Studer et al., 1998)- *"An ontology is a formal, explicit specification of a shared conceptualization"*. Ontology engineering is a complex process. We followed a systematic approach presented in (de Almeida Falbo, 2014) to build an ontology for evolution provenance of scientific workflows. This method considers activities for the development of reference ontologies and to its implementation as operational ontologies.

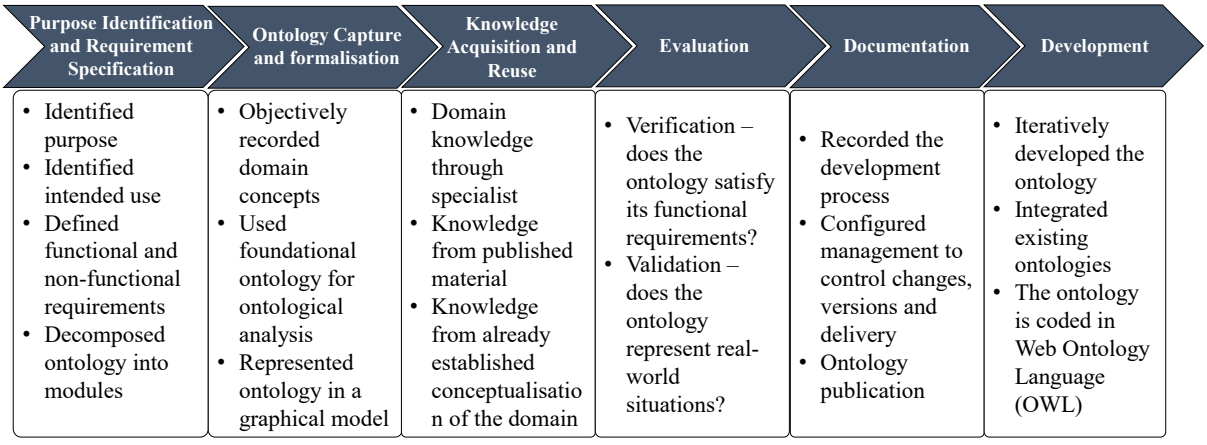

Fig. 1. Steps of the WePROV development process

The ontology development process includes six steps, as illustrated in Figure 1. In **Identification and Requirements Elicitation**, we identify the purpose and intended uses of the ontology, define its functional requirements through Competency Questions (CQs), and non-functional requirements (NFRs),

[8] https://research.csiro.au/digiscape/

and decompose the ontology into appropriate modules. During the **Ontology Capture and Formalisation** phase, we conceptualise the domain using a foundation ontology and represent it in a graphic model. Also, for **Knowledge Acquisition**, we gather domain knowledge reliably through specialists and published material and take advantage of conceptualisations already established for the domain. For **Evaluation** purposes, we evaluate the suitability of the ontology through *verification*, ensuring that the ontology satisfies its requirements, and *validation*, ensuring that the ontology can represent real-world situations. Further, the **Documentation** stage records the results of the development process using a Reference Ontology Specification followed by configuration management, which is performed through a repository to control changes, versions, and delivery. Ontology **Development** is an iterative process where each version is evaluated over the generic and domain-specific competency questions and iteratively improved by assessing its ability to answer the competency questions. After designing the ontology, we create instance repositories for workflow evolutions provenance and can answer the competency questions via SPARQL queries.

3.2. Ontology Requirement Specification

We closely collaborated with the developers of scientific workflow engines that includes WorkSpace (Cleary et al., 2015; Bolger et al., 2016) and Senaps[9] and provenance researchers from the CSIRO[10], the Australian National University[11], and SurroundAustralia[12] to elicit the following functional and non-functional requirements for the WePROV ontology.

3.2.1. Non-functional Requirements

We identified the following non-functional requirements:

- **NFR1:** Be modular or embedded in a modular framework to facilitate reuse of other ontologies and, consequently, its reuse by other ontologies.
- **NFR2:** Be based on well-known sources from the literature.
- **NFR3:** Be documented in a standard-compliant and machine-readable format.

To address **NFR1**, we decompose the ontology into two modules namely, WePROV-workflow and WePROV-evolution. The former defines the components and structure of a workflow (Section 4.1), and the latter defines the the main concepts and relationships to describe workflow evolution process (Section 4.2). As part of NFR1, we integrated WePROV into the Provenance Ontology (PROV-O) and the ProvONE ontology. The prefixes of these reused ontologies in WePROV are preceded by the corresponding acronyms (prov: and provone: respectively), throughout the paper. Regarding **NFR2**, ontology capture is supported by the process of knowledge acquisition that used consolidated sources of knowledge referring to the scientific workflows in this research, including books and standards. Regarding **NFR3**, ontology is documented in a machine-readable format i.e., RDF/OWL.

3.2.2. Functional Requirements

The authors of this paper worked with Senaps team for more than twelve months to identified a series of tasks that need to be supported by a useful workflow evolution provenance ontology as the functional requirements. Next, the competency questions (CQs) are defined iteratively based on the identified tasks. Here, we list eleven of the most important CQs to evaluate the proposed ontology.

[9] https://products.csiro.au/senaps/
[10] https://www.csiro.au/
[11] https://www.anu.edu.au/
[12] https://surroundaustralia.com/

- **CQ1: Find the temporal reconstruction of the workflow design.** Users want to understand the evolution of the research (i.e., workflow) since inception to date. The ontology should be able to provide information about different versions of the workflow, how often the workflow has evolved and at what point on time. An indication of the exact changes is also desirable.
- **CQ2: Identify different changes in a version of the workflow.** Users require to know if the contents or structures of the workflow have evolved to understand the rationale behind the changes. The ontology should support queries to identify all changes.
- **CQ3: Report the reason(s) of divergent results of two executions of a workflow.** Users are curious to identify the root cause of the unexpected behaviour of the workflow. One such reason could be the change in workflow, or it could be some accidental changes in the workflow. Comparing two versions pinpoint the changed composition of the workflow.
- **CQ4: Which one is the most unstable component of the workflow?** Users are interested to know a component that is the reason of frequent changes in the workflow. The ontology should support the identification of the component that has frequent revisions.
- **CQ5: Identify all the agents who have participated in the design of the workflow.** The ontology should be able to identify all participants who contributed to any version of the workflow instead of those who contributed to the latest/last version.
- **CQ6: Who is responsible for a change in the workflow?** Users need to know who is responsible for a change in the workflow. This requirement is a typical auditing question that assigns responsibility to a participant for the change made.
- **CQ7: Who participated in a specific (first/latest) version of the workflow?** Users need to attribute the work to an agent. This attribution information is useful to track down the issues or to give proper credit to the original participants.
- **CQ8: Find all workflows (or versions of a workflow) where an agent has participated.** This question addresses the need to identify the work/research of an agent and its participation. The ontology should support queries to find workflows, versions, and components where an agent has participated along with the time of participation.
- **CQ9: Find the collaborative researchers and their collaboration count of designing workflows together.** Users need to know researchers (i.e., agents) who most often collaborate and their count of collaborative workflows. This information could help in identifying collaborative communities who have similar research interests.
- **CQ10: When a (first/last) version of the workflow is created?** Users need to know when the workflow was designed and since when it has not changed. The ontology should be able to record the effect of time on workflow lifespan.
- **CQ11: How long did an agent participate in a workflow (or repository of a workflow)?** Users need to identify the period a specific agent was active in the lifespan of a workflow. The ontology should answer the start and end time of an agent's participation.

3.3. Knowledge Acquisition and Reuse

It is crucial to understand the existing provenance ontologies for their reuse to design a workflow evolution provenance ontology. First, we consider PROV-DM, a conceptual data model that forms a basis for the W3C provenance ontology PROV-O in Section 3.3.1. Next, we revisit some existing workflow provenance ontologies and evaluate their reusability to capture workflow evolution provenance in Section 3.3.2.

3.3.1. PROV-DM: Baseline provenance model

PROV provides a generic data model, PROV-DM[13], to outline the provenance. It is a W3C recommendation and designed to be an agnostic model for provenance description from different areas. The PROV data model can represent data transformations and ownership, and it can be extended to fulfil the requirements of domains. The classes *Entity*, *Activity*, and *Agent* along with the relationships *Used*, *WasGeneratedBy*, *WasInformedBy*, *WasAssociatedWith*, *WasDerivedFrom*, *WasAttributedTo*, and *ActedOnBehalfOf* form the core model of PROV-DM as shown in Figure 2. The second component of PROV-DM is dealing with derivations of entities from other entities and derivation subtypes *Revision*, *Quotation*, and *Primary Source*.

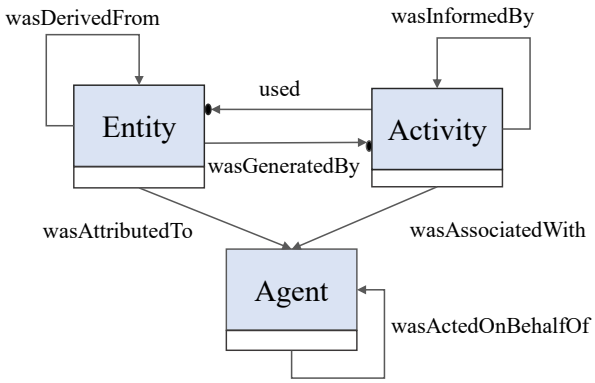

Fig. 2. PROV Core Structure

PROV is developed to promote an interoperable interchange of provenance in heterogeneous environments, such as the Web. However, it is a generic and domain-independent ontology, as it does not cater to the requirements of specific systems or domain applications. Instead, it provides extension points through which such systems and applications can extend PROV for their domain. Moreover, it is quite easy to come up with anyone's own provenance ontology for a domain. However, the heterogeneous models to capture provenance of a domain of interest defeat the entire purpose of a community effort and thus breaks the interoperability, which is the central premise of PROV. Therefore, a standard provenance ontology is necessary for a domain to meet the interoperability goal of the Web.

3.3.2. Scientific workflow model

In recent years, the Semantic Web community has proposed provenance ontologies to formalise and describe scientific workflows and their execution traces and has shown several applications of these ontologies (Oliveira et al., 2018). In this regard, Wf4Ever (Belhajjame et al., 2015) project addresses challenges related to generically describing scientific experiments through the *wfdesc* ontology, and an analysis and management of their execution provenance through the *wfprov* ontology. The Open Provenance Model for Workflows (OPMW) (Garijo and Gil, 2011) presents a framework to publish computational workflows including the specification of the *OPMW* ontology, for the description of workflow traces and their templates. As part of this framework, the authors also published the Ontology for Provenance and Plans (P-Plan)[14]. P-Plan extends the Provenance Ontology (PROV-O) (Moreau and Missier, 2013)

[13] https://www.w3.org/TR/prov-dm/
[14] http://www.opmw.org/model/p-plan/

to specify the plans that guide the execution of scientific processes, describing how such plans are composed and their correspondence to provenance records that describe the execution. ProvONE (Cuevas-Vicenttín et al., 2016) is another provenance ontology to describe scientific workflow, which is compatible with PROV-DM. It provides constructs to describe workflow specification and workflow execution provenance. Each of these ontologies can capture, store, and query the provenance of workflows and their traces in a common, machine-readable, and workflow-engine independent format. However, they are unable to describe workflow evolution provenance. ProvONE supports a minimal level of evolution provenance. For instance, for a workflow shown in Figures 3 and 4, Listing 1 represents ProvONE aligned evolution provenance.

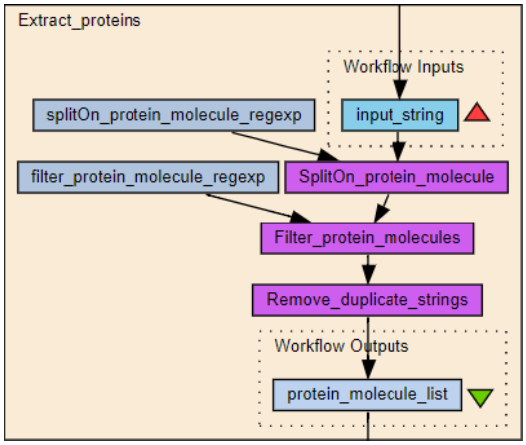

Fig. 3. Extract_Proteins_v1

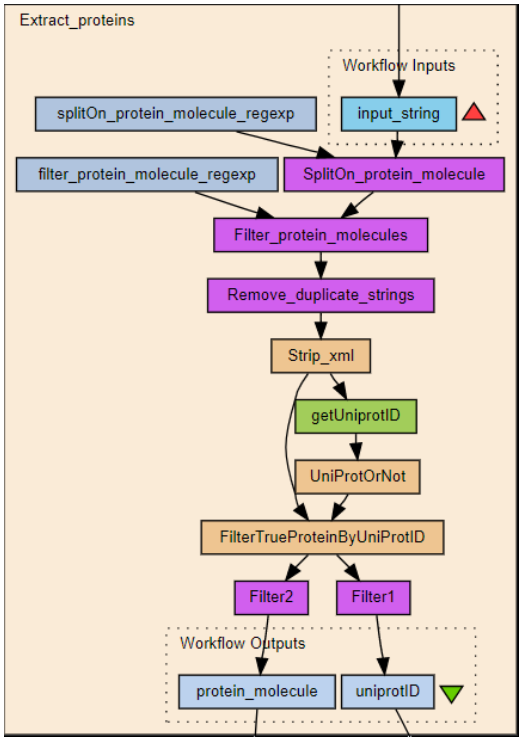

Fig. 4. Extract_Proteins_v2

Here, the versions of a workflow form a derivation tree, which is represented with *wasDerivedFrom* association from PROV-O ontology. The evolution provenance captures that "Extract_Proteins_v2" is derived from "Extract_Proteins_v1". However, the specific changes performed in the specification of a workflow are uncaptured in ProvONE aligned evolution provenance.

Listing 1: ProvONE aligned evolution provenance

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix provone: <http://purl.org/provone> .
@prefix : <http://example.com/> .
```

```
:Extract_Proteins_v2
      rdf:type provone:Workflow;
      dcterms:identifier "v2"^^xsd:string;
      dcterms:title "Extract_Proteins"^^xsd:string;

:Extract_Proteins_v2 prov:wasDerivedFrom :Extract_Proteins_v1
```

In Section 4, we will provide a different perspective of evolution provenance, which *wasDerivedFrom* association cannot capture.

4. WePROV: Workflow Evolution Provenance Modelling

To fulfil the need for workflow evolution provenance, we propose a **WePROV** - **W**orkflow **e**volution **PROV**enance ontology in this paper.

4.1. WePROV-Workflow

In this regard, we need to formalise the structure and components of a workflow that can evolve and understand the provenance ontologies mentioned above for their reusability. ProvONE is a widely used workflow provenance ontology because of its ability to capture comprehensive scientific workflows provenance (Prabhune et al., 2018). We reuse workflow specification pattern shown in Figure 5 from ProvONE and provide an additional layer to capture workflow evolution provenance.

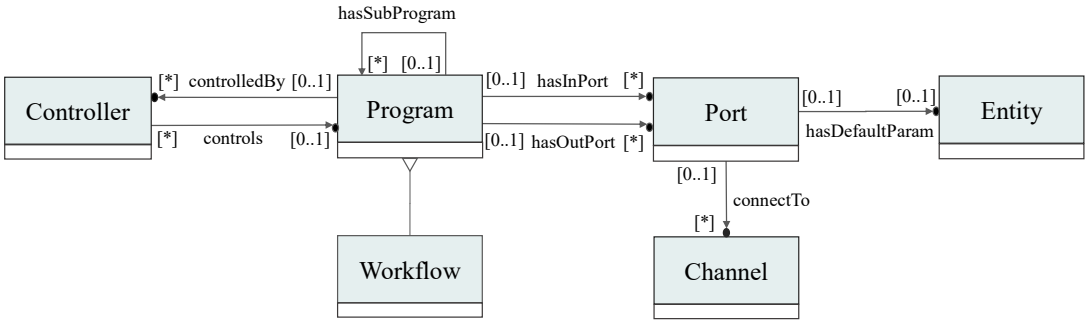

Fig. 5. Scientific Workflow Specification Model

Figure 5 highlights the key components of a workflow, which can evolve and to which evolution provenance applies.

- **Program** ($\mathcal{P}$)**:** The various tasks that are part of a workflow are represented by the Program class. Programs can either be atomic or composite, the latter case specified through the *hasSubProgram* self-association. A workflow itself is a program.
- **Port:** Each Program may have a series of Ports that function as input ($\mathcal{I}$) or output ($\mathcal{O}$) ports.
- **Channel** ($\mathcal{C}$)**:** Ports of the Programs connect through Channels. Both input and output ports may associate with multiple Channels. Connecting a single port to multiple channels capture workflows where a single output is copied and sent to multiple destinations. Moreover, it models workflows where tasks take inputs from different sources through a single input port.

- **Parameter ($\mathcal{A}$):** In order to specify executable instances of a Workflow, default parameters can be defined for some of its constituent programs. The default parameters are represented by Entities.
- **Controller ($\mathcal{L}$):** A Controller class can be used to specify that the execution of a given program is controlled by another program, which allows for differing model of executions. For instance, in the asynchronous data flow model, a given Program may only start once the execution of a preceding program terminates.

Based on the elements of a workflow, we defined our workflow as:

Definition 4.1. *A workflow $\mathcal{W}$ is a graph that connects programs $\mathcal{P}$ to data $\mathcal{D}$ and other programs in the graph through edges $\mathcal{E}$.*

$$\mathcal{W} = (\mathcal{P}, \mathcal{D}, \mathcal{E}) \tag{1}$$

Where $(\mathcal{I} \cup \mathcal{O} \cup \mathcal{A}) \in \mathcal{D}$ and $(\mathcal{L} \cup \mathcal{C}) \in \mathcal{E}$

4.2. WePROV-Evolution

Three phases of workflow evolution are considered within our approach: workflow creation, workflow deletion, and workflow modification, as shown in Figure 6. Workflow creation deals with two main activities that are related and, in some cases, overlap but are fundamentally different activities. A workflow can be created by creating new models and specifying a new workflow structure, or by merging and reusing existing workflows. The former is referred to workflow creation and the latter is referred to workflow derivation in this paper. In Definition 4.1, we defined a workflow with a graph in which nodes and edges correspond to operational tasks, and data flows between these tasks respectively. Therefore, two types of modification activities that affect the workflow are distinguished: structural modification (addition or deletion of nodes or edges); and content modification (modification of a node or an edge). Sections 4.2.1, 4.2.2 and 4.2.3 provide a detailed discussion on workflow evolution.

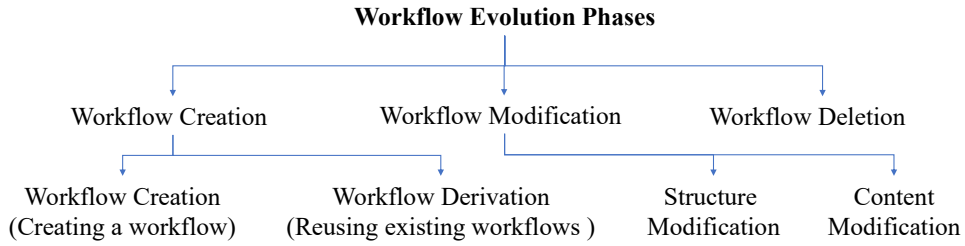

Fig. 6. Workflow evolution phases

4.2.1. Creation of the Workflow

Scientific workflows can be designed using a workflow engine. They are often not completely disjoint, and some of them may reuse other workflows. Consequently, the creation of a workflow can be done either by creating all elements of the workflow from scratch or by reusing the existing workflows.

- **Workflow Creation:** When a workflow is designed, all its elements are created during workflow creation time. There is no derivation history for the workflow. We used generation pattern to model the workflow creation provenance.
Figure 7 shows the provenance pattern for workflow creation. For brevity, `Workflow` here represents the workflow and its elements. `Generation` is the completion of the production of a new workflow by an activity `Creation`. This workflow does not exist before generation and becomes available for use after this generation. `Agent` is the creator of this workflow.

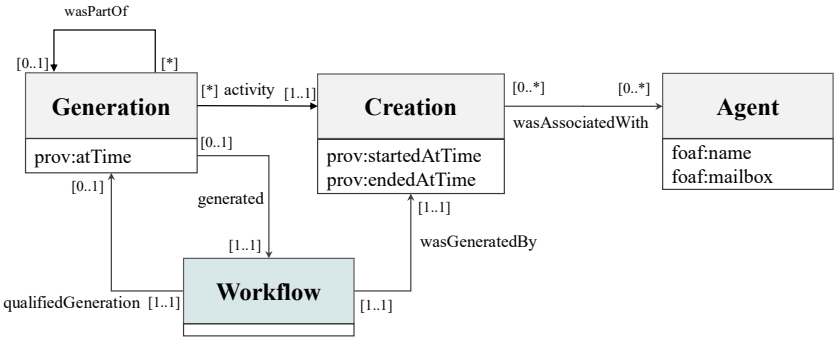

Fig. 7. Workflow Creation Provenance Pattern

In creation provenance, workflow and its constituent elements will have their dedicated generation instances; therefore, multiple generations under the same creation activity may occur. However, in this case, the generation of the elements would be part of the generation of the workflow, as shown in axioms A1 and A2.

$$\forall\, x_w, x_e : \text{Generation}, \forall\, c : \text{Creation}, \text{activity}(x_w, c) \wedge \text{activity}(x_e, c) \rightarrow \text{wasPartOf}(x_e, x_w) \textbf{ (A1)}$$

$$\forall\, x_w, x_e : \text{InstantaneousEvent}, \text{Generation}(x_e) \wedge \text{wasPartOf}(x_e, x_w) \rightarrow \text{Generation}(x_w) \textbf{ (A2)}$$

- **Workflow Derivation:** A workflow can be created by reusing an existing workflow referred to as workflow derivation in this paper. A workflow must exist and execute in its entirety so that it can be reused while creating another workflow.
Figure 8 shows the provenance model for workflow creation when it is derived from another workflow. `Derivation` constructs a new workflow based on a pre-existing workflow. A `workflow` uses another workflow during the `creation` of the workflow in such a derivation. In its purest form, derivation relates to two workflows where the `usage` of the `usedEntity` is provided to make the derivation path explicit.
Information such as `creation` and `usage` are linked to `derivation`, as shown in axiom A3, to aid analysis of provenance.

$$\forall\, w_i, w_j : \text{Workflow}, \forall\, d : \text{Derivation}, \forall\, u : \text{Usage}, \text{qualifiedDerivation}(w_i,d) \wedge \text{hadUsage}(d,u) \wedge \text{usedEntity}(u,w_k) \rightarrow \text{wasDerivationOf}(w_i,w_j) \textbf{ (A3)}$$

Moreover, the derivation is transitive, i.e.,

$$\forall\, w_i, w_j, w_k : \text{Workflow}, \text{wasDerivationOf}(w_i,w_j) \wedge \text{wasDerivationOf}(w_j,w_k) \rightarrow \text{wasDerivationOf}(w_i,w_k) \textbf{ (A4)}$$

Finally, for all workflows there should be either a derivation or a generation provenance record i.e.,

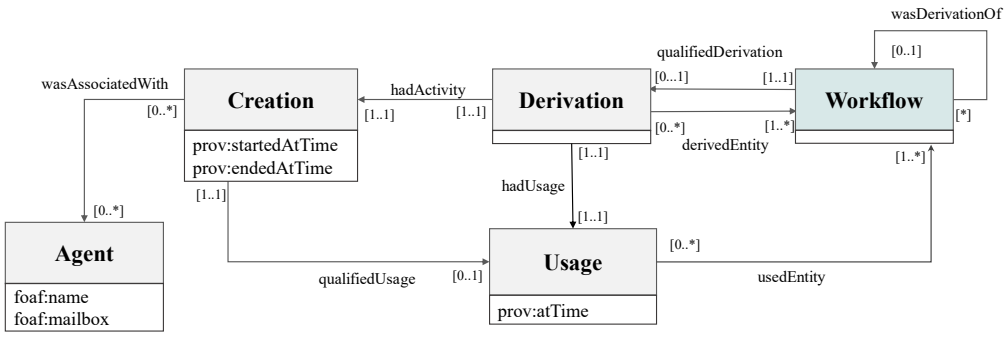

Fig. 8. Workflow Derivation Provenance Pattern

$$\forall\, w : \text{Workflow}, \exists\, c \mid c : \text{Creation} \wedge (\ \text{wasGeneratedBy}(w,c) \oplus (\ \text{qualifiedDerivation}(\text{w,d}) \wedge \text{hadActivity}(\text{d,c}))\ \textbf{(A5)}$$

4.2.2. Deletion of the Workflow

The suppression or deletion of a workflow is handled by invalidation of that workflow. This makes the workflow unavailable for further use, but it is accessible to explore the evolution of a workflow. We used invalidation concept from PROV-DM to model a workflow deletion provenance pattern. Figure 9 shows the provenance pattern for workflow deletion. For brevity, *Workflow* here represents the workflow and(or) its elements. `Invalidation` is the start of the cessation or expiry of an existing workflow by a `Deletion` activity. The workflow is no longer available for the use after invalidation. Any generation or usage of a workflow precedes its invalidation. A `Deletion` is associated with an `Agent`.

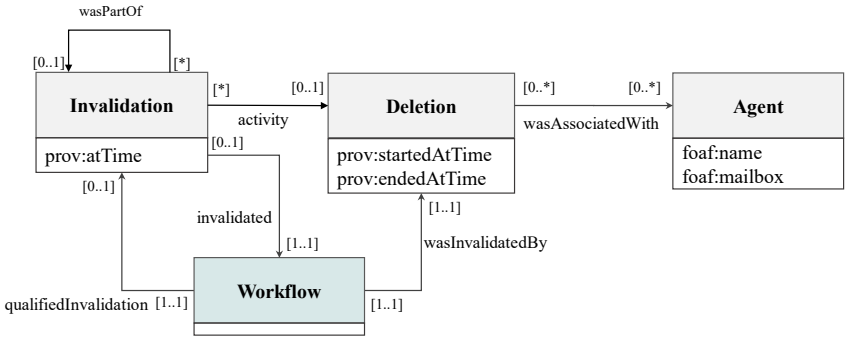

Fig. 9. Workflow Invalidation Provenance Pattern

When a workflow is invalidated, all its elements are invalidated as a part of the workflow deletion activity, as described in A5.

$$\forall\, x_i, x_j : \textit{Invalidation}, \forall\, w : \text{Workflow}, \forall\, p : \text{Program} \cup \text{Controller} \cup \text{Port} \cup \text{Channel} \cup \text{Parameter}, p \in w \wedge \text{qualifiedInvalidation}(w, x_i) \rightarrow \text{qualifiedInvalidation}(p, x_j) \wedge \text{wasPartOf}(x_j, x_i)\ \textbf{(A5)}$$

The workflow and its elements have their dedicated invalidation instances and the invalidation event of elements is a part of workflow invalidation, as shown in axioms A6 and A7.

$$\forall\, x_i, x_j : \text{InstantaneousEvent}, \text{Invalidation}(x_i) \wedge \text{wasPartOf}(x_j, x_i) \rightarrow \text{Invalidation}(x_i)\ \textbf{(A6)}$$

$$\forall\, x_i, x_j : \text{Invalidation}, \forall\, d : \text{Deletion}, \text{activity}(x_i, d) \wedge \text{activity}(x_j, d) \rightarrow \text{wasPartOf}(x_i, x_j) \vee \text{wasPartOf}(x_j, x_i) \quad \textbf{(A7)}$$

4.2.3. *Modification of the Workflow*

Workflows are likely to be modified due to different reasons, and there is a need to manage the modification provenance.

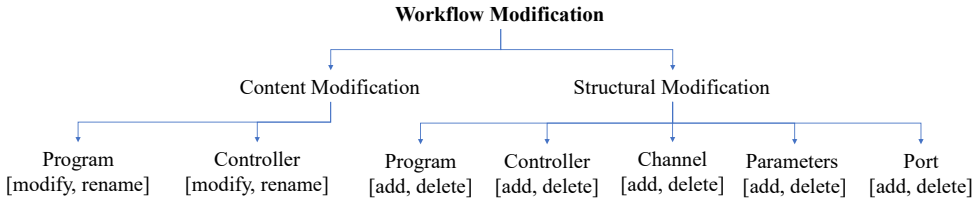

Fig. 10. Possible Modification types in Workflows

Two types of modifications might affect a workflow i.e., a change in workflow structure or its contents. Modifications in workflow structure includes the addition or suppression of a Program, Controller, Channel, Parameter, or Port. Whereas, modification in workflow contents includes modification of a Program. Moreover, a workflow can be modified as a result of a single change in the workflow or composite changes, which may include structural and content changes in a modification activity. A hierarchy of possible modifications in a workflow is designed considering the given aspects, as shown in Figure 10. This figure shows the elements of a workflow and type of operations (i.e., add, delete, modify and rename) that are relevant with reference to workflow modification.

Here, addition and deletion of a workflow or its elements (i.e., Program, Controller, Channel, or Parameter) represent a structural modification. Whereas, modifying and renaming these components are content modifications. Moreover, modifying a Program includes modifying or renaming a model (i.e., Program content modification) or adding, deleting, and renaming Ports (i.e., Program interface modification).

A workflow modification provenance pattern is presented in Figure 11 based on the possible modifications in workflows.

A `Revision` is a derivation of a workflow for which the resultant workflow is a revised version of some original workflow and contains substantial content from the original. Revision relates two workflows where a revised workflow is the latest version of the workflow. The implication here is that revision results in the generation of a new version of the workflow while invalidating the original one. This new version of the workflow is the *current* version. While the *previous* version is not deleted, rather it becomes invalidated. To provide a completely accurate description of the revision, the `Generation` and `Invalidation` of the generated and invalidated entities are provided. We then make the derivation path explicit through activity, generation, and invalidation. Linking such information with revision aids analysis of provenance and facilitates moving backward and forward in revised workflows history.

Revision is transitive just like derivation, i.e.,

$$\forall\, w_i, w_j, w_k : \text{Workflow}, \text{wasRevisionOf}(w_i,w_j) \wedge \text{wasRevisionOf}(w_j,w_k) \rightarrow \text{wasRevisionOf}(w_i,w_k) \quad \textbf{(A8)}$$

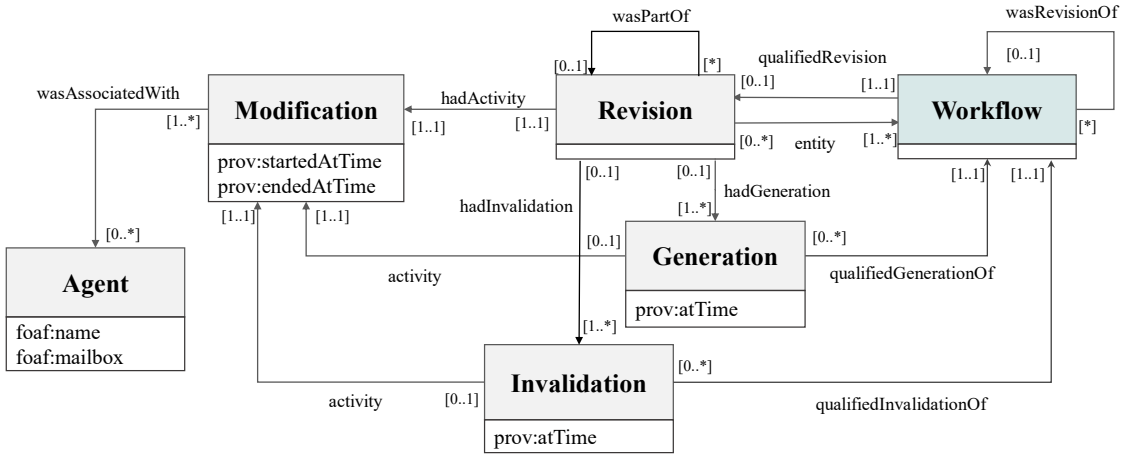

Fig. 11. Workflow Modification Provenance Pattern

A workflow revision represents the composite changes in a workflow. However, a composite change (workflow modification) is an aggregation of atomic changes, i.e., modification in the elements (programs, controllers, channels, and parameters), as shown in Figure 10. While a composite change is captured through revision pattern shown in Figure 11, atomic changes can be captured differently based on the modification operator. For example,

- *Addition:* adding a component in a workflow during a revision is captured by **creation pattern** shown in Figure 7.
- *Deletion:* Our interest is in managing the evolution of a workflow; therefore, instead of deleting a workflow or its component the suppression or deletion is represented with **invalidation pattern** as shown in Figure 9.
- *Modification:* modifying a component is a revision of original component in a workflow, therefore, is captured by the **modification pattern** shown in Figure 11. A component revision is envisioned as a part of a workflow revision.
- *Renaming:* renaming is captured through the same **modification pattern**, however, the activity that handles renaming will distinguish it from modification.

4.3. Workflow Evolution Provenance Ontology

In previous sections, we considered PROV as a baseline and built an extension that accommodates the need for workflow evolution-based provenance. The corresponding OWL representation of the WePROV ontology is available here[15]. Now, we present the key classes and relationships, which form the **W**orkflow **E**volution **Prov**enance Ontology. The proposed ontology has a workflow-centric perspective and revolves around the creation, deletion, modification, and renaming of workflows and its different elements. However, the modelling of workflows evolution aligns with the PROV-O ontology.

Figure 12 shows the ontology structure underlying all the four modelling perspectives. The colour of the classes and properties represent their participation in one or more patterns presented in Sections 4.2.1, 4.2.2, and 4.2.3. In this figure, **workflow** represents a complete workflow or its elements.

[15] https://github.com/anilabutt/WePROV-Workflow-Evolution-Provenance-Ontology/blob/master/Ontology/weprov.owl

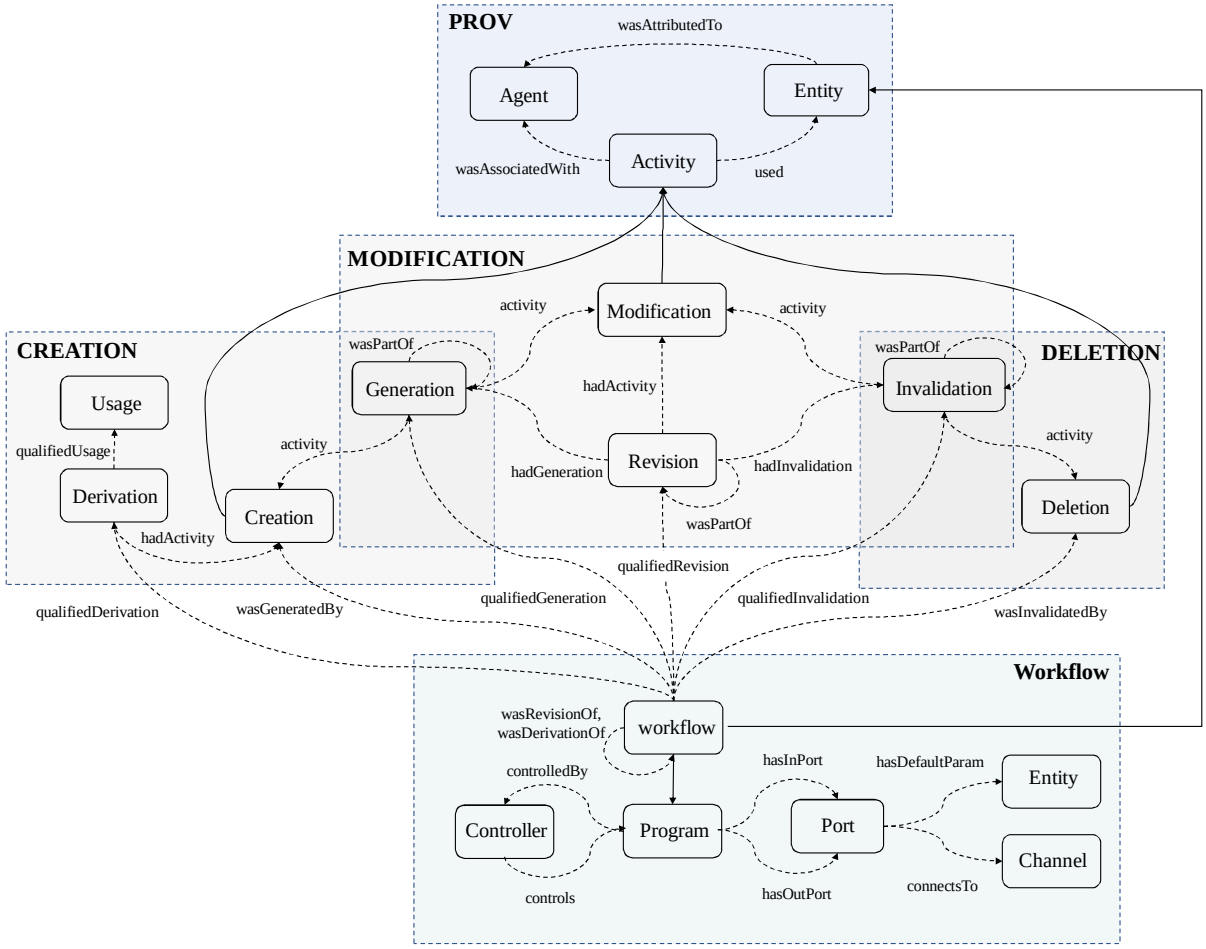

Fig. 12. The workflow evolution model and its relationship to PROV

The activities of **creation**, **deletion**, **modification** and **renaming** participates in the evolution by **generation**, **usage**, and **invalidation** of workflows and its components, each performed by an **agent** and is started and ended at a time. **Derivation** uses some workflows to generate some new workflows, whereas **revision** invalidates elements of workflows to generate their new versions.

5. Evaluation

The evaluation of the ontology comprises activities of *validation* and *verification* by ensuring that the ontology can represent real-world situation and satisfies their requirements.

5.1. WePROV Verification

For ontology verification, SABiO (de Almeida Falbo, 2014) suggests identifying whether the elements that make up the ontology can answer the raised competency questions. For verification, we implemented WePROV in formal ontological language OWL which allows creating SPARQL queries for

testing our ontology by instantiating the competency questions on realistic examples. Listing 2 presents the SPARQL query for CQ4. The SPARQL queries for the raised competency questions (cf. Sect. 3.2.2), showing which concepts and relations are used to answer a CQ is available here[16].

Listing 2: Which is the most unstable component of the workflow?

```
PREFIX provone:<http://purl.dataone.org/provone/2015/01/15/ontology#>
PREFIX prov:<http://www.w3.org/ns/prov#>
PREFIX weprov:<http://www.csiro.au/digiscape/weprov#>

CONSTRUCT {?workflowId weprov:hasRevisions ?revisions}
FROM <http://weprov.csiro.au/evolution/>
WHERE {
        SELECT ?workflowId ( MAX(?revCount) AS ?revisions)
        FROM <http://weprov.csiro.au/>
        FROM <http://weprov.csiro.au/evolution/>
        WHERE {
                ?modification a weprov:Modification.
                ?revision a prov:Revision; weprov:revision ?revCount; prov:activity
                    ↪ ?modification; prov:wasRevisionOf ?workflowId.
                FILTER (?workflowId = ?workflow) {
                        SELECT ?workflow
                        WHERE { <workflowURI> (provone:hasSubProgram/
                            ↪ provone:hasSubProgram)* ?workflow.} }
        } GROUP BY (?workflowId)
}
```

5.2. WePROV Validation

According to SABiO, a valid ontology should have the ability to properly represent real-world situations. For validation, we created WePROV aligned open-source workflow provenance dataset using scientific workflows accessible through a shared platform called myExperiment (De Roure et al., 2009) The platform provides a service[17] to register scientific workflows, where $\sim$ 2850 such workflows have been registered (March 2020). All workflows on *myExperiment* have a version history ranging from 1 $\sim$ 17 versions per workflow. About 2100 (i.e., 75%) workflows cover life sciences experiments, designed using Taverna (Hull et al., 2006). About $\sim$ 570 (i.e., 25%) Taverna workflows are documented using *scufl* data model, which is the preliminary version of Taverna's existing data model[18]. Taverna1 workflow dataset[19] is the next biggest publically available workflows dataset and in-use relevant designed format, is chosen to validate our ontology. Our WePROV aligned dataset[20] has a collection of workflows which were retrieved by crawling a seed set of workflow Ids derived from myExperiment platform along with their specification and evolution provenance derived using WePROV model.

Listings 3, 4, and 5 show some examples of WePROV instantiation. Listing 3 is an example of workflow specification provenance for *BioAIDDiseaseDiscovery*[21] workflow, which finds disease relevant to

[16] https://github.com/anilabutt/WePROV-Workflow-Evolution-Provenance-Ontology/blob/master/CompetencyQuestions/SPARQL-for-CQs.txt

[17] https://www.myexperiment.org/workflow

[18] http://ns.taverna.org.uk/2010/scufl2#

[19] https://www.myexperiment.org/workflows/?filter=TYPE_ID%28%221%22%29

[20] https://github.com/anilabutt/WePROV-Workflow-Evolution-Provenance-Ontology/tree/master/Dataset

[21] https://www.myexperiment.org/workflows/72/versions/1.html

the query string. The workflow specification pattern describes a workflow, its inputs, outputs, processes and links. It is noteworthy that as it is a complete description of the workflow, the specification can be used for machine enabled workflow analytics and interoperability.

Listing 3: BioAIDDiseaseDiscovery workflow specification provenance

```
@prefix weprov: <http://www.csiro.au/digiscape/weprov#>.
@prefix provone: <http://purl.dataone.org/provone/2015/01/15/ontology#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix : <http://weprov.csiro.au/> .

<workflow/618ac202-acf6-4695-bdc6-ca0078be3649-v0>
   a provone:Workflow ;
   provone:hasInPort :query_string ;
   provone:hasOutPort :relevant_documents, :discovered_proteins, :discovered_diseases ;
   provone:hasSubProgram [a provone:Program, weprov:Stringconstant ;
      provone:controlledBy :Document_index.value-Retrieve_documents.document_index;
      rdfs:label "Document_index" .] ,
      [a provone:Program, weprov:Stringconstant;
      provone:controlledBy :search_field.value-Retrieve_documents.search_field ;
      rdfs:label "search_field" .] ,
      [a provone:Program , weprov:Stringconstant ;
      provone:controlledBy :maxHits.value-Retrieve_documents.maxHits ;
      rdfs:label "maxHits" .] ,
      [a provone:Program , weprov:Beanshell ;
      provone:controlledBy :Remove_xml_tag.term-Link_proteins_to_diseases.keyword ;
      rdfs:label "Remove_xml_tag" .] ,
      [ a provone:Program , weprov:Workflow ;
      provone:controlledBy :Flatten_and_make_unique.flattened_unique_output-
          ↪ discovered_diseases ;
      provone:hasSubProgram [a provone:Workflow;
         provone:hasInPort :input ;
         provone:hasOutPort :flattened_unique_output ;
         provone:hasSubProgram :Remove_duplicate_strings , :Flatten_list ;
         dcterms:description "..." ;
         dcterms:title "Flatten_and_make_unique" ;
         provone:hasController :input-Flatten_list.inputlist, :Flatten_list.outputlist:
             ↪ Remove_duplicate_strings.stringlist, :Remove_duplicate_strings.strippedlist-
             ↪ flattened_unique_output ;
         weprov:revision "0" .] ;
      rdfs:label "Flatten_and_make_unique" .] ,
      [a provone:Program , weprov:Workflow ;
      provone:controlledBy :Link_proteins_to_diseases.OMIM_disease_label-
          ↪ Flatten_and_make_unique.input ;
      provone:hasSubProgram [a provone:Workflow ;
         provone:hasInPort :keyword ;
         provone:hasOutPort :OMIM_disease_label ;
         provone:hasSubProgram :Remove_duplicate_strings, :Flatten_list, :
             ↪ filter_disease_regexp, :split_OMIM_regexp, :Extract_diseases_from_OMIM, :
             ↪ search, :Split_OMIM_results, :label_OMIM_disease ;
         dcterms:description "..." ;
         dcterms:title "Link_protein_to_OMIM_disease" ;
         provone:hasController :Flatten_list.outputlist-Remove_duplicate_strings.stringlist, :
             ↪ keyword-search.keyword, :Extract_diseases_from_OMIM.filteredlist-
             ↪ label_OMIM_disease.OMIM_disease_string, :Remove_duplicate_strings.strippedlist
             ↪ -OMIM_disease_label, :Split_OMIM_results.split-Extract_diseases_from_OMIM.
             ↪ stringlist, :filter_disease_regexp.value-Extract_diseases_from_OMIM.regex, :
             ↪ label_OMIM_disease.OMIM_disease_label-Flatten_list.inputlist, :search.Result-
             ↪ Split_OMIM_results.string, :split_OMIM_regexp.value-Split_OMIM_results.regex ;
```

```
        weprov:revision "0" .] ;
     rdfs:label "Link_proteins_to_diseases" .],
     [a provone:Program , weprov:Workflow ;
     provone:controlledBy :Discover_proteins.discovered_proteins-Remove_xml_tag.tagged_term,
         ↪ :Discover_proteins.discovered_proteins-discovered_proteins ;
     provone:hasSubProgram [a provone:Workflow ;
        provone:hasInPort :documents_from_lucene ;
        provone:hasOutPort :discovered_proteins ;
        provone:hasSubProgram :prelearned_genomics_model, :Extract_proteins , :
            ↪ Discover_entities ;
        dcterms:description "This workflow applies the discovery workflow ..." ;
        dcterms:title "Discover_proteins" ;
        provone:hasController :documents_from_lucene-Discover_entities.input_from_lucene, :
            ↪ Discover_entities.discovered_entities-Extract_proteins.input_string, :
            ↪ prelearned_genomics_model.value-Discover_entities.learned_model, :
            ↪ Extract_proteins.protein_molecule_list-discovered_proteins ;
        weprov:revision "0" .] ;
     rdfs:label "Discover_proteins" .] ,
     [a provone:Program , weprov:Workflow ;
     provone:controlledBy :Retrieve_documents.relevant_documents-Discover_proteins.
         ↪ documents_from_lucene, :Retrieve_documents.relevant_documents-relevant_documents
         ↪ ;
     provone:hasSubProgram [a provone:Workflow ;
        provone:hasInPort :query_string>, :document_index, :search_field, :maxHits ;
        provone:hasOutPort :relevant_documents ;
        provone:hasSubProgram :Biooptimize_query, :Retrieve ;
        dcterms:description "This workflow retrieves relevant documents..." ;
        dcterms:title "Retrieve_bio_documents" ;
        provone:hasController :query_string-Biooptimize_query.query_string, :
            ↪ Biooptimize_query.optimized_lucene_query-Retrieve.queryString, :document_index
            ↪ -Retrieve.document_index, :maxHits-Retrieve.maxHits, :search_field-Retrieve.
            ↪ search_field, :Retrieve.relevant_documents-relevant_documents ;
        weprov:revision "0" .] ;
     rdfs:label "Retrieve_documents" .] ;
  dcterms:description "This workflow finds disease relevant..." ;
  dcterms:title  "BioAID_DiseaseDiscovery" ;
  provone:hasController  :query_string-Retrieve_documents.query_string, :Discover_proteins.
      ↪ discovered_proteins-Remove_xml_tag.tagged_term, :Document_index.value-
      ↪ Retrieve_documents.document_index, :Link_proteins_to_diseases.OMIM_disease_label-
      ↪ Flatten_and_make_unique.input, :Remove_xml_tag.term-Link_proteins_to_diseases.
      ↪ keyword, :Retrieve_documents.relevant_documents-Discover_proteins.
      ↪ documents_from_lucene, :maxHits.value-Retrieve_documents.maxHits, :search_field.
      ↪ value-Retrieve_documents.search_field, :Discover_proteins.discovered_proteins-
      ↪ discovered_proteins, :Flatten_and_make_unique.flattened_unique_output-
      ↪ discovered_diseases, :Retrieve_documents.relevant_documents-relevant_documents ;
  weprov:revision "0" .
```

When a workflow is designed for the first time, the creation pattern of WePROV is instantiated being the only relevant phase of workflow evolution. Listing 4 presents an overview of *workflow creation provenance* for *"BioAIDDiseaseDiscovery"* workflow. It is noteworthy that creation provenance records the act, time, and agent of the generation of a workflow and its sub-workflows. Also, generation of the elements of a workflow, e.g., ports, programs and controllers are captured as part of the creation provenance.[22] This provenance information can answer the queries: Who was the initiator of a workflow? What elements constituted the first version of the workflow? And when the workflow was created for the first time?

[22] This part of creation provenance is skipped in Listing 4 for brevity.

Listing 4: BioAIDDiseaseDiscovery workflow creation provenance

```
@prefix weprov: <http://www.csiro.au/digiscape/weprov#> .
@prefix provone: <http://purl.dataone.org/provone/2015/01/15/ontology#> .
@prefix prov:<http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix : <http://weprov.csiro.au/> .

<evolution/generation/618ac202-acf6-4695-bdc6-ca0078be3649>
      a prov:Generation ;
      prov:activity [a weprov:Creation , prov:Activity;
            prov:wasAssociatedWith [a prov:Agent ;
                  foaf:name "Marco Roos (AID)" .] .
            ] ;
      prov:generated [a provone:Workflow;
            dcterms:title "BioAID_DiseaseDiscovery" ;
            weprov:version "0"];
      prov:atTime "2020-02-11T04:48:10.6Z"^^xsd:dateTimeStamp .

<evolution/generation/f43db36c-a3ed-4f78-8d1c-89f27dfb53f7>
      a prov:Generation ;
      prov:activity [a weprov:Creation , prov:Activity;
            prov:wasAssociatedWith [a prov:Agent ;
                  foaf:name "Marco Roos (AID)" .] .
            ] ;
      prov:generated [a provone:Workflow;
            dcterms:title "Flatten_and_make_unique" ;
            weprov:version "0".];
      weprov:wasPartOf <evolution/generation/618ac202-acf6-4695-bdc6-ca0078be3649>;
      prov:atTime "2020-02-11T04:48:10.528Z"^^xsd:dateTimeStamp .

<evolution/generation/b4c1a118-6a38-40b5-99e9-febbd3c85f2b>
      a prov:Generation ;
      prov:activity [a weprov:Creation , prov:Activity;
            prov:wasAssociatedWith [a prov:Agent ;
                  foaf:name "Marco Roos (AID)" .] .
            ] ;
      prov:generated [a provone:Workflow;
            dcterms:title "Link_protein_to_OMIM_disease" ;
            weprov:version "0".] ,
      weprov:wasPartOf <evolution/generation/618ac202-acf6-4695-bdc6-ca0078be3649>;
      prov:atTime "2020-02-11T04:48:10.559Z"^^xsd:dateTimeStamp .

<evolution/generation/dd1e2961-a1ca-4902-9bfb-2b776a4399ee>
      a prov:Generation ;
      prov:activity [a weprov:Creation , prov:Activity;
            prov:wasAssociatedWith [a prov:Agent ;
                  foaf:name "Marco Roos (AID)" .] .
            ] ;
      prov:generated [a provone:Workflow;
            dcterms:title "Discover_proteins" ;
            weprov:version "0".] ,
      weprov:wasPartOf <evolution/generation/618ac202-acf6-4695-bdc6-ca0078be3649>;
      prov:atTime "2020-02-11T04:48:10.59Z"^^xsd:dateTimeStamp .

<evolution/generation/4dccdaac-5994-4350-b30b-28eac86c229a>
      a prov:Generation ;
      prov:activity [a weprov:Creation , prov:Activity;
            prov:wasAssociatedWith [a prov:Agent ;
                  foaf:name "Marco Roos (AID)" .] .
```

```
            ] ;
      prov:generated [a provone:Workflow;
            dcterms:title "Retrieve_documents" ;
            weprov:version "0".] ,
      weprov:wasPartOf <evolution/generation/618ac202-acf6-4695-bdc6-ca0078be3649>;
      prov:atTime "2020-02-11T04:48:10.531Z"^^xsd:dateTimeStamp .
```

The revision pattern of the ontology is instantiated to capture the changes for each subsequent revision of the workflow. To instantiate the revision component of WePROV, we have provided an example of the *"Extract_Proteins"* workflow, which is a sub-workflow of our example workflow, i.e., *"BioAID-DiseaseDiscovery"*. Figures 3 and 4 show the initial (i.e., *"Extract_Proteins_v1"*) and the revised (i.e., *"Extract_Proteins_v2"*) workflows, respectively.

The WePROV captures revision in-terms of Δ_iW, which is the difference between any two successive versions of the workflow, as shown in Equation 2.

$$\Delta_i W = W_{(i+1)} - W_{(i)} \tag{2}$$

Here Δ_iW captures all the changes introduced in the latest version. Listing 5 represents WePROV aligned evolution provenance. The provenance captures $\Delta_1 \mathrm{W}_{ExtractProtein}$ i.e., all the changes made in *Extract_Proteins_v2*. The provenance here depicts that two ports, six programs, and nine controllers are added, and a port and a controller has been removed.

Listing 5: Extract_Proteins workflow revision provenance

```
@prefix weprov: <http://www.csiro.au/digiscape/weprov#> .
@prefix provone: <http://purl.dataone.org/provone/2015/01/15/ontology#> .
@prefix prov:<http://www.w3.org/ns/prov#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix : <http://weprov.csiro.au/> .

<evolution/revision/1/df6063f9-b469-4d56-aecc-a62db4bcb3ad>
   a prov:Revision ;
   prov:revisedEntity [a provone:Program , weprov:Workflow ;
     provone:hasSubProgram [a provone:Workflow;
       dcterms:title "Extract_proteins" ;
       dcterms:description "This workflow filters protein molecule labeled terms from an input
           ↪ string...";
       weprov:revision "1" .] ;
     rdfs:label "Extract_proteins" .];
   weprov:wasPartOf [a prov:Revision
     prov:revisedEntity [a provone:Program , weprov:Workflow ;
       rdfs:label "Discover_RatHumanMouseUniProt_proteins" .];
   prov:activity [a weprov:Modification , prov:Activity ;
     prov:startedAtTime "2020-02-11T04:59:27.699Z"^^xsd:dateTimeStamp ;
     prov:wasAssociatedWith :Marco_Roos_(AID)> .];
   prov:atTime "2020-02-11T04:59:53.846Z"^^xsd:dateTimeStamp ;
   prov:hadGeneration [a prov:Generation;
     prov:qualifiedGenerationOf [a provone:Port;
       rdfs:label "protein_molecule".],
       [a provone:Port;
       rdfs:label "uniprotID".],
```

```
        [a provone:Program, weprov:Beanshell;
        rdfs:label "Strip_xml".],
        [a provone:Program, weprov:Arbitrarywsdl;
        rdfs:label "getUniprotID".],
        [a provone:Program, weprov:Beanshell;
        rdfs:label "UniProtOrNot".],
        [a provone:Program, weprov:Beanshell;
        rdfs:label "FilterTrueProteinByUniProtID".],
        [a provone:Program, weprov:Local;
        rdfs:label "Filter1".],
        [a provone:Program, weprov:Local;
        rdfs:label "Filter2".],
        [a provone:Controller;
        rdfs:label "Remove_duplicate_strings.strippedlist-Strip_xml.tagged_term".],
        [a provone:Controller;
        rdfs:label "Strip_xml.term-getUniprotID.term".],
        [a provone:Controller;
        rdfs:label "Strip_xml.term-FilterTrueProteinByUniProtID.protein".],
        [a provone:Controller;
        rdfs:label "getUniprotID.getUniprotIDReturn-UniProtOrNot.uniprotIDlist".],
        [a provone:Controller;
        rdfs:label "UniProtOrNot.uniprotID_or_False-FilterTrueProteinByUniProtID.uniprot".],
        [a provone:Controller;
        rdfs:label
        "FilterTrueProteinByUniProtID.true_uniprot-Filter1.stringlist".],
        [a provone:Controller;
        rdfs:label "FilterTrueProteinByUniProtID.true_protein-Filter2.stringlist".],
        [a provone:Controller;
        rdfs:label "Filter1.filteredlist-uniprotID".],
        [a provone:Controller;
        rdfs:label "Filter2.filteredlist-protein_molecule".].
    ];
  prov:hadInvalidation [a prov:Invalidation;
    prov:qualifiedInvalidationOf [a provone:Port;
      rdfs:label "protein_molecule_list".],
      [a provone:Controller;
      rdfs:label "Remove_duplicate_strings.strippedlist-protein_molecule_list".].
   ] .
```

This provenance information can be used to *RollBack* or *Undo* from the latest workflow version without storing the previous versions of a workflow. Equation 3 shows the *RollBack* process using WePROV aligned provenance:

$$W_{(i)} = W_{(i+1)} - \Delta_i W \tag{3}$$

"Extract_Proteins_v1" can be generated from *"Extract_Proteins_v2"* in case we inverse the changes introduced in the latest version. For example, all components generated and invalidated in $\Delta_1 W_{ExtractProtein}$ are invalidated and generated in *"Extract_Proteins_v2"*, respectively. The act of reversing $\Delta_i W$ will undo the changes and roll back the workflow to its previous version. Similarly, the revision provenance directs in *RollForward* or *Redo* to a latest workflow version from a previous version. Equation 4 shows the *RollForward* process using WePROV aligned provenance:

$$W_{(i+1)} = W_{(i)} + \Delta_i W \tag{4}$$

Finally, the results of the validation and verification were checked carefully to assess the correctness and completeness for the running example. Also, two of the provenance domain experts who are familiarised with the details of the WePROV confirmed the added value of the conceptualisation based on the instantiated dataset. One of the key points is that the ontology helps to capture workflow provenance as it better describes the workflow evolution than PROV-O or ProvONE constructs. The evaluation shows that our designed ontology facilitates workflow evolution applications to perform analyses and provides more intuitive ways for exploration, including the ability to return to a previous workflow version.

6. Previous Work

Workflow provenance has been studied in a variety of domains, including experimental science, business, and data analytic (Herschel et al., 2017). However, the topic of workflow evolution provenance is less focussed compared to workflow specification or execution provenance. As far as we know, the principal work done in this field has been carried out in *VisTrails* system (Freire et al., 2006; Koop et al., 2010). VisTrails provides visual interfaces to produce visualisations by assembling pipelines out of modules connected in a network. It captures and maintains a detailed provenance of the visualisation process, including the evolution of a visualisation workflow. Each evolution is considered as a new version of the workflow, and it has a unique id, name, optional annotation, set of actions, and a set of macros. Moreover, each action is uniquely identified by a timestamp (@time), which corresponds to the time the action was executed. Although the system captures the evolution of workflows, the underlined evolution provenance model is limited in capturing all workflow evolution phases discussed in this paper. Also, the solutions rely on a proprietary format that makes interchanging provenance information difficult.

Kepler (Altintas et al., 2006) records provenance information in a relational schema and provides access to the information through Java interface. The provenance schema[23] records when specifying the workflow structure for the first time, or when parameter values change. *Triana* (Majithia et al., 2004) provenance data recorded includes date and time composed, date and time of the execution, details of the resource on which a service was executed (i.e., input data), and intermediate data products generated. Triana did not record the workflow evolution provenance except the time and date when a workflow was composed. *Taverna* (Oinn et al., 2004) captures workflow execution provenance, which is used to populate previous runs and intermediate results in the *Taverna Workbench Results perspective*[24]. The provenance trace can be exported as a PROV-O RDF graph which can be queried using SPARQL and processed with other PROV tools, such as the PROV Toolbox[25]. However, it does not capture provenance of editing a workflow definition (workflow evolution), but assume the scientist manages the evolution of workflow through existing means for versioning files, such as filenames and folders, version control systems like git, or workflow sharing websites like myExperiment. *WINGS/Pegasus* (Kim et al., 2008) supports workflow specification and execution provenance to answer the provenance queries supported by the framework. However, it focuses on the workflow execution provenance on the cloud, rather than the provenance of the workflow itself (e.g., design changes). *Galaxy* (Goecks et al., 2010) tracks provenance to ensure repeatability of the experimental results. The provenance information includes input

[23] https://code.kepler-project.org/code/kepler/trunk/modules/provenance/docs/provenance.pdf
[24] http://www.taverna.org.uk/documentation/taverna-2-x/provenance/
[25] https://lucmoreau.github.io/ProvToolbox/

datasets, tools used, parameter values, and output datasets. However, this system also misses workflow evolution provenance.

Moreover, there exist stand-alone approaches for workflow evolution provenance. However, most of the solutions lack their ability in managing the evolution provenance in one or another way. *Carvalho et al.* (Carvalho et al., 2018) proposed a framework to help scientists in exploring local adjustments to a workflow by replacing software and software versions used to implement a workflow component, however, it cannot trace workflow evolution process. *Workflow Evolution Framework* (Barga et al., 2010; Withana et al., 2010) includes a versioning model to manage workflow evolution. The authors considered two dimensions of workflow evolution. One where a workflow is evolved due to some change in structure or component and second where an existing component is reused in the other workflow. However, only author, time and version number are all the metadata recorded for an evolved workflow. *Lins et al.* (Lins et al., 2008) presented some initial study on how workflow evolution provenance can be used to derive some useful statistics from analysing new aspects of workflow specification and design. However, the authors did not propose any workflow evolution provenance management strategy.

7. Conclusion

This paper presents an ontology to describe a workflow and its evolution (i.e., change) in a generic, machine-readable and interoperable format. Existing provenance models and ontologies are unable to describe the evolution of a workflow and result in limited workflow provenance analytics. To overcome the limitation, we introduced WePROV- a workflow evolution provenance ontology, which complements the existing provenance ontologies to capture the evolution of scientific workflows and enables automatic tracking and traversing among different versions of the workflow. WePROV is built following the SABiO ontology engineering method and is based on well-known ontologies like PROV and ProvONE. It supports several operations to explore different versions of a workflow, including the ability to return to a previous workflow version in an intuitive way. We evaluated the suitability of the ontology through rigorous verification and validation. For validation purpose, we generated a real dataset, which is a collection of workflows retrieved from the myExperiment platform with their specification and evolution provenance derived using WePROV ontology. Moreover, we implemented WePROV in a formal ontological language OWL, which allows creating SPARQL Queries to verify our ontology by instantiating the competency questions on real world scenarios. In future, we intend to use the ontology at the foundation of a RESTful web service, which provides a provenance management solution for scientific workflows. Also, we will use the ontology in the context of semantic interoperability among different workflow management systems.

References

Altintas, I., Barney, O. & Jaeger-Frank, E. (2006). Provenance collection support in the kepler scientific workflow system. In *International Provenance and Annotation Workshop* (pp. 118–132). Springer.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B. & Mock, S. (2004). Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* (pp. 423–424). IEEE.

Auer, S. & Herre, H. (2006). RapidOWL—An agile knowledge engineering methodology. In *International Andrei Ershov Memorial Conference on Perspectives of System Informatics* (pp. 424–430). Springer.

Barga, R.S., Simmhan, Y.L., Chinthaka, E., Sahoo, S.S., Jackson, J. & Araujo, N. (2010). Provenance for Scientific Workflows Towards Reproducible Research. *IEEE Data Eng. Bull.*, *33*(3), 50–58.

Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J.M., Bechhofer, S., et al. (2015). Using a suite of ontologies for preserving workflow-centric research objects. *Journal of Web Semantics*, *32*, 16–42.

Bolger, M., Cleary, P., Cohen, R., Harrison, S., Hetherton, L., Rucinski, C., Sankaranarayanan, N., Thomas, D., Watkins, D. & Zhang, Z. (2016). Workspace: a fast and low cost methodology for delivering commercial applications based on Research IP. *Proc. eResearch Australasia*.

Butt, A.S., Car, N. & Fitch, P. (2020). Towards Ontology Driven Provenance in Scientific Workflow Engine. In *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*. MODELSWARD 2020 (pp. 105–115).

Cao, Y., Jones, C., Cuevas-Vicenttín, V., Jones, M.B., Ludäscher, B., McPhillips, T., Missier, P., Schwalm, C., Slaughter, P., Vieglais, D., et al. (2016). DataONE: a data federation with provenance support. In *International Provenance and Annotation Workshop* (pp. 230–234). Springer.

Carvalho, L.A.M., Garijo, D., Medeiros, C.B. & Gil, Y. (2018). Semantic Software Metadata for Workflow Exploration and Evolution. In *2018 IEEE 14th International Conference on e-Science (e-Science)* (pp. 431–441). IEEE.

Cleary, P., Thomas, D., Bolger, M., Hetherton, L., Rucinski, C. & Watkins, D. (2015). Using Workspace to automate workflow processes for modelling and simulation in engineering. In 'MODSIM2015. In *21st International Congress on Modelling and Simulation', Broadbeach, Queensland, Australia* (pp. 669–675).

Cohen-Boulakia, S., Belhajjame, K., Collin, O., Chopard, J., Froidevaux, C., Gaignard, A., Hinsen, K., Larmande, P., Le Bras, Y., Lemoine, F., et al. (2017). Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Generation Computer Systems*, *75*, 284–298.

Cuevas-Vicenttín, V., Ludäscher, B., Missier, P., Belhajjame, K., Chirigati, F., Wei, Y., Dey, S., Kianmajd, P., Koop, D., Bowers, S., et al. (2016). ProvONE: A PROV Extension Data Model for Scientific Workflow Provenance (2015). https://purl.dataone.org/provone-v1-dev. [Online; accessed 16-May-2019].

de Almeida Falbo, R. (2014). SABiO: Systematic Approach for Building Ontologies. In *ONTO. COM/ODISE@ FOIS*.

de Almeida Falbo, R., de Menezes, C.S. & da Rocha, A.R.C. (1998). A systematic approach for building ontologies. In *Ibero-American Conference on Artificial Intelligence* (pp. 349–360). Springer.

De Roure, D., Goble, C. & Stevens, R. (2009). The design and realisation of the Experimentmy Virtual Research Environment for social sharing of workflows. *Future Generation Computer Systems*, *25*(5), 561–567.

Deelman, E., Gannon, D., Shields, M. & Taylor, I. (2009). Workflows and e-Science: An overview of workflow system features and capabilities. *Future generation computer systems*, *25*(5), 528–540.

Deelman, E., Peterka, T., Altintas, I., Carothers, C.D., van Dam, K.K., Moreland, K., Parashar, M., Ramakrishnan, L., Taufer, M. & Vetter, J. (2018). The future of scientific workflows. *The International Journal of High Performance Computing Applications*, *32*(1), 159–175.

Freire, J., Silva, C.T., Callahan, S.P., Santos, E., Scheidegger, C.E. & Vo, H.T. (2006). Managing rapidly-evolving scientific workflows. In *International Provenance and Annotation Workshop* (pp. 10–18). Springer.

Garijo, D. & Gil, Y. (2011). 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 (pp. 47–56). ACM. doi:10.1145/2110497.2110504.

Goecks, J., Nekrutenko, A. & Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, *11*(8), R86.

Gómez-Pérez, A., Fernández, M. & De Vicente, A. (1996). Towards a method to conceptualize domain ontologies. In *ECAI96 Workshop on Ontological Engineering, Budapest* (pp. 41–51).

Haller, A., Marmolowski, M., Gaaloul, W., Oren, E., Sapkota, B. & Hauswirth, M. (2009). From Workflow Models to Executable Web Service Interfaces. In *2009 IEEE International Conference on Web Services* (pp. 131–140). doi:10.1109/ICWS.2009.51.

Herschel, M., Diestelkämper, R. & Ben Lahmar, H. (2017). A survey on provenance: What for? What form? What from? *The VLDB Journal—The International Journal on Very Large Data Bases*, *26*(6), 881–906.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P. & Oinn, T. (2006). Taverna: a tool for building and running workflows of services. *Nucleic acids research*, *34*(suppl_2), W729–W732.

Kim, J., Deelman, E., Gil, Y., Mehta, G. & Ratnakar, V. (2008). Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, *20*(5), 587–597.

Koop, D., Scheidegger, C.E., Freire, J. & Silva, C.T. (2010). The provenance of workflow upgrades. In *International Provenance and Annotation Workshop* (pp. 2–16). Springer.

Kradolfer, M. & Geppert, A. (1999). Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In *Proceedings Fourth IFCIS International Conference on Cooperative Information Systems. CoopIS 99 (Cat. No. PR00384)* (pp. 104–114). IEEE.

Lins, L., Koop, D., Anderson, E.W., Callahan, S.P., Santos, E., Scheidegger, C.E., Freire, J. & Silva, C.T. (2008). Examining statistics of workflow evolution provenance: A first study. In *International Conference on Scientific and Statistical Database Management* (pp. 573–579). Springer.

Majithia, S., Shields, M., Taylor, I. & Wang, I. (2004). Triana: A graphical web service composition and execution toolkit. In *Proceedings. IEEE International Conference on Web Services, 2004.* (pp. 514–521). IEEE.

Moreau & Missier (2013). World Wide Web Consortium "PROV-DM: The PROV Data Model" W3C Recommendation . https://www.w3.org/TR/prov-dm/. [Online; accessed 26-August-2019].

Noy, N.F., McGuinness, D.L., et al. (2001). Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and ....

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., et al. (2004). Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, *20*(17), 3045–3054.

Oliveira, W., Ambrósio, L.M., Braga, R., Ströele, V., David, J.M. & Campos, F. (2017). A framework for provenance analysis and visualization. *Procedia Computer Science*, *108*, 1592–1601.

Oliveira, W., Oliveira, D.D. & Braganholo, V. (2018). Provenance analytics for workflow-based computational experiments: A survey. *ACM Computing Surveys (CSUR)*, *51*(3), 53.

Prabhune, A., Zweig, A., Stotzka, R., Hesser, J. & Gertz, M. (2018). P-PIF: a ProvONE provenance interoperability framework for analyzing heterogeneous workflow specifications and provenance traces. *Distributed and Parallel Databases*, *36*(1), 219–264.

Roddick, J.F. (1995). A survey of schema versioning issues for database systems. *Information and Software Technology*, *37*(7), 383–393.

Sacha, D., Kraus, M., Bernard, J., Behrisch, M., Schreck, T., Asano, Y. & Keim, D.A. (2017). Somflow: Guided exploratory cluster analysis with self-organizing maps and analytic provenance. *IEEE transactions on visualization and computer graphics*, *24*(1), 120–130.

Stitz, H., Luger, S., Streit, M. & Gehlenborg, N. (2016). Avocado: Visualization of workflow–derived data provenance for reproducible biomedical research. In *Computer Graphics Forum* (Vol. 35, pp. 481–490). Wiley Online Library.

Studer, R., Benjamins, V.R. & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data knowledge engineering*, *25*(1-2), 161–197.

Taylor, I., Shields, M., Wang, I. & Harrison, A. (2007). The triana workflow environment: Architecture and applications. In *Workflows for e-Science* (pp. 320–339). Springer.

Van Der Aalst, W.M. & Ter Hofstede, A.H. (2005). YAWL: yet another workflow language. *Information systems*, *30*(4), 245–275.

Withana, E.C., Plale, B., Barga, R. & Araujo, N. (2010). Versioning for workflow evolution. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (pp. 756–765). ACM.