Towards Ontology Driven Provenance in Scientific Workflow Engine

Anila Sahar Butt[1][a], Nicholas Car[2][b] and Peter Fitch[1][c]
[1]*CSIRO Land and Water, Canberra, Australia*
[2]*SURROUND Australia Pty Ltd, Brisbane, Australia*
{*anila.butt, peter.fitch*}*@csiro.au, nicholas.car@surroundaustralia.com*

Keywords: Workflow Provenance, Provenance Ontology, Provenance from Event Logs.

Abstract: Most workflow engines automatically capture and provide access to their workflow provenance, which enables its users to trust and reuse scientific workflows and their data products. However, the deed of instrumenting a workflow engine to capture and query provenance data is burdensome. The task may require adding hooks to the workflow engine, which can lead to perturbation in execution. An alternative approach is intelligent logging and a careful analysis of logs to extract critical information about workflows. However, rapid growth in the size of the log and the cloud-based multi-tenant nature of the engines has made this solution increasingly inefficient. We have proposed ProvAnalyser, an ontology-based approach to capture the provenance of workflows from event logs. Our approach reduces provenance use cases to SPARQL queries over captured provenance and is capable of reconstructing complete data and invocation dependency graphs for a workflow run. The queries can be performed on nested workflow executions and can return information generated from one or various executions.

1 INTRODUCTION

The Oxford English Dictionary defines provenance as *"the source or origin of an object; its history and pedigree; a record of the ultimate derivation and passage of an item through its various owners."* In the context of computer applications, provenance is an essential component to allow for result reproducibility, sharing, and knowledge reuse for different stakeholders. It facilitates the users in interpreting and understanding results by examining the sequence of steps that led to a result (Curcin, 2017).

With the realisation of data-driven science, scientists are increasingly adopting workflows to specify and automate repetitive experiments that retrieve, integrate, and analyse datasets to produce scientific results (Belhajjame et al., 2015). In recent years, the scientific community has developed various scientific workflow engines to provide an environment for specifying and enacting workflows (e.g., Taverna, Kepler, Daliuge, and Airflow). Among those, **Senaps**[1] is a custom build workflow engine designed through the need of hosting applications from multiple domains (e.g., marine sensing, water management, and agriculture). The focus of Senaps is on hosting, adapting, and sharing existing scientific models or analysis code across organisations and groups who use the sensor, climate, and other time-series data.

Due to the dynamic nature of the platform, Senaps must consider its workflow provenance, which concerns the reliability and integrity of workflows and their potentially complex data processes. Understanding workflow provenance is crucial for Senaps users to identify bottlenecks, inefficiencies, learn how to improve them, and trust in data produced by these workflows. Moreover, to gain an understanding of a workflow, and how it may be used and reproduced for their needs, scientists require access to additional resources, such as annotations describing the workflow, datasets used and produced by this workflow, and provenance traces recording workflow executions. With the realisation of the value, provenance can bring to the overall architecture of Senaps; its development team is planning to integrate a provenance collection and querying component into Senaps.

Senaps can integrate provenance component, which is an elegant solution but requires a significant effort to implement. It requires adding hooks to Senaps architecture to capture provenance data, which can lead to perturbation in execution. Therefore, the Senaps team decided to thoroughly under-

[a] https://orcid.org/0000-0002-3508-6049
[b] https://orcid.org/0000-0002-8742-7730
[c] https://orcid.org/0000-0002-9813-0588
[1] https://research.csiro.au/dss/research/senaps/

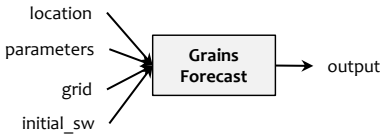

Figure 1: An abstract workflow to forecast a grain production on a location.

stand the provenance capturing, storage, and access requirements before integrating the provenance solution into Senaps to avoid any performance issues. Moreover, the team needed to comprehend the means of collecting provenance of already executed workflows using their event log. Since workflow provenance is event-based, i.e., capturing the significant events within a system, the event log is an essential source of provenance data. Logs are traditionally used for auditing and identifying the root causes of failure in large systems. However, logs also contain essential information about the events within a system that result in the generation of data objects. It has been shown that intelligent logging and careful analysis of logs support to extract critical information about the system (Oliner and Stearley, 2007). Currently, the team answers the provenance related queries through the analysis of workflows and their execution traces using their event log. However, the rapid growth in the size of the event log and the cloud-based multi-tenant nature of the platform has made this solution increasingly inefficient.

In this paper, we show our work capturing workflow provenance from event logs of Senaps. For a workflow, as shown in Figure 1, we would like to: (a) enable scientists and developers to ask questions about a workflow run by providing convenient queries against the captured provenance traces; (b) have the engine track the exact data dependencies within a run so that answers to such scientific questions may be as accurate as possible. For this, we present **ProvAnalyser**[2], an ontology-based approach for provenance capturing and querying system for Senaps. It transforms Senaps event logs into knowledge graphs using an ontology that supports a set of provenance queries. Our approach reduces provenance use cases to SPARQL[3] queries over the knowledge graph, and is capable of reconstruction complete data and invocation dependency graphs for a workflow run. In this regard, we:

- detail the design of **SENProv** – an ontology to model provenance data of Senaps workflows specification and execution with the main goal of empowering provenance data to be interoperable.
- show capturing raw provenance data from the event log, their mapping to **SENProv**, and the storage of structured provenance data in a database.
- describe the design and running of provenance use cases[4] to analyse the impact of provenance on Senaps and its client applications.

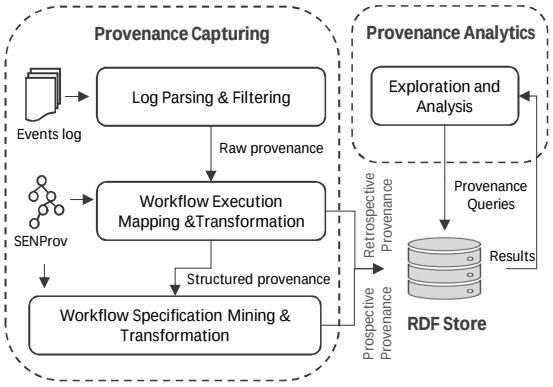

Figure 2: The ProvAnalyser Architecture.

The rest of the paper is organised as follows. In Section 2, we discuss ProvAnalyser in terms of its architecture, information model (SENProv ontology), and its provenance extraction, structuring, and storage mechanisms. In Section 3, we present the provenance use cases and their results. In Section 4, we discuss the steps required for large-scale deployment of the technology within the organisation. In Section 5, we review state-of-the-art and in Section 6, we conclude outlining future directions of research and development.

2 ProvAnalyser

Figure 2 shows the architecture of our proposed approach. ProvAnalyser captures provenance from Senaps event logs containing the workflows' event execution traces and stores it in an RDF store. On this stored provenance data, one can perform analysis and exploration through predefined provenance queries. It works as follows:

1. For each workflow execution request, all traces related to that request are parsed and provenance information (i.e., execution time, workflowId, oper-

[2]https://github.com/CSIRO-enviro-informatics/ProvAnalyser

[3]https://www.w3.org/TR/sparql11-query/

[4]We store all the use cases and the corresponding SPARQL queries developed for this work in the code repository https://github.com/CSIRO-enviro-informatics/senprov-usecases

atorNodeId, model, ports, and data nodes) is filtered. It transforms a verbose event log into concise raw provenance data.

2. The raw provenance data is mapped to SENProv and is transformed into a structured provenance for that particular execution trace. It generates the RDF[5] description of retrospective provenance (Herschel et al., 2017) (i.e., workflow execution) and stores in an RDF store.
3. It infers prospective provenance (Herschel et al., 2017) (i.e., workflow structure) from the retrospective provenance extracted from event log using SENProv. It then links the retrospective provenance associated with the prospective provenance and stores both in the RDF store.
4. ProvAnalyser allows users to explore and analyse provenance by designing provenance use cases, running them as SPARQL queries over RDF store, and displaying the results to their clients.

ProvAnalyser supports a range of provenance use cases, such as explaining and reproducing the outcome of a workflow, tracing the effect of a change, and provenance analytics. It provides a structure to provenance information, which makes it machine-readable and interoperable. Therefore, the provenance data can also be used and integrated with other provenance solutions. Moreover, it reduces the time needed for analysing workflow execution traces and allows semantic web experts to perform the task, thus distributing the load.

2.1 Workflows in Senaps

The UML diagram in Figure 3 represents the conceptual model for a workflow specification and execution in Senaps. We confirmed the model during a meeting (Joe and Charman 2018, personal communication, 13 September).

Here **workflow** is a multi-directed acyclic graph[6] made up of vertices and edges, which are referred as nodes and connections in Senaps. A node can either be a **data node** or an **operator node**. An operator node hosts a model (executable code and its supporting files). The operator node has multiple **ports**, whereas a data node can only connect to an operator node through a port. Currently, a data node supports multi-stream, document, and grid data formats A user **group** or an **organisation** put a workflow execution request. With a **workflow execution** request, a **user** needs to specify the workflow to execute, the data

[5]https://www.w3.org/RDF/

[6]https://en.wikipedia.org/wiki/Directed_acyclic_graph

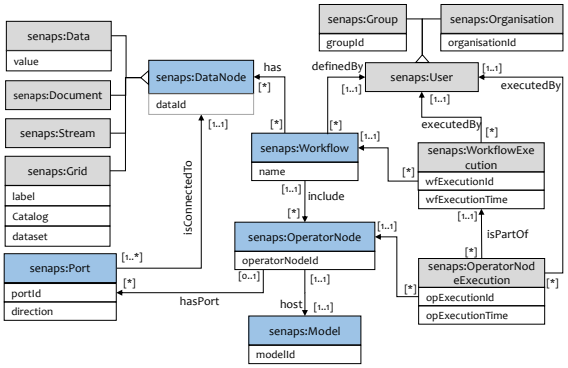

Figure 3: Senaps Workflow Specification (constructs in blue) and Execution (constructs in grey) Conceptual Model UML Diagram.

node (i.e., input data), and the port on which a data node is connecting to an operator node. A workflow execution id is assigned to the run when it executes. Each operator node of the workflow is executed and has its operator node execution id, and corresponding input and output data nodes. Therefore, one workflow execution is composed of all its operator nodes executions.

2.2 Provenance Ontology

To capture the provenance of Senaps workflows, we require a data model capable of capturing all the metadata (i.e., Senaps constructs) shown in Figure 3. Some generic and extendable provenance models already exist in the literature to capture data and(or) workflow provenance.

PROV-DM is the World Wide Web Consortium (W3C)-recommended data model for the interoperable provenance in heterogeneous environments, such as the Web (Moreau and Missier, 2013). PROV-DM is generic and domain-independent and does not cater to the specific requirements of particular systems or domain applications; rather, it provides extension points through which systems and applications can extend PROV-DM for their purposes.

However, Senaps is concerned with capturing provenance from complex computational pipelines commonly referred to as scientific workflows. Several recent community efforts have culminated with the development of generic models to represent the provenance of scientific workflows. We have evaluated ProvONE, OPMW, and Wf4Ever as the most expressive of these models (Oliveira et al., 2018a) for their capability to reuse for the design of a data model for Senaps. *OPMW* (Garijo and Gil, 2011) is a conceptual model for the representation of prospective and retrospective provenance collected

Table 1: Senaps Constructs Mapping to ProvONE and PROV-DM Constructs.

Senaps Aspect	Construct Type	Senaps Concept	ProvONE/ PROV-DM
Workflow	Class	Workflow	provone:Workflow
		OperatorNode	provone:Program
		DataNode	provone:Channel
		Port	provone:Port
		Model	prov:Plan
	Property	include	provone:has-SubProgram
		hasPort	provone:hasInPort provone:hasOutPort
		isConnectedTo	provone:connectTo
Workflow Execution	Class	WorkflowExecution	provone:Execution
		OperatorNode-Execution	provone:Execution
		Organisation	provone:User
		Group	provone:User
		Document	prov:Entity
		Stream	prov:Entity
		Grid	prov:Entity
	Property	initiatedBy	provone:agent prov:wasAssociat-edWith
		isPartOf	provone:wasPartOf
		wfExecutionTime	prov:atTime
		opExecutionTime	prov:atTime
		value	prov:value

from the execution of scientific workflows. It is a specialisation of PROV and the OPM provenance model. *Wf4Ever* (Belhajjame et al., 2015) has extended PROV to present wfdesc and wfprov ontologies for the description of prospective and retrospective provenance respectively. ProvONE (Cuevas-Vicenttín et al., 2016) is a data model, built on PROV-DM, for scientific workflow provenance representation. It provides constructs to model workflow specification provenance (i.e., a set of instructions specifying how to run a workflow) and workflow execution provenance (i.e., the record of how the workflow is executed). ProvONE is a widely accepted workflow provenance model and is capable of capturing all the characteristics shown in Figure 3; therefore, we specialise ProvONE in SENProv to capture the provenance of Senaps workflows. SENProv takes an event-centric perspective and revolves around workflow specification and workflow execution events.

To reuse ProvONE and PROV-DM in **SENProv**, we need to model the relationship of Senaps constructs shown in Figure 3 with PROM-DM and ProvONE constructs. Table 1 shows the mapping between Senaps and PROV-DM or ProvONE. In SENProv, each Senaps class shown in *'Senaps Concept'* column extends from its corresponding class presented in *'ProvONE/PROV-DM'* column of the table, and ProvONE or PROV-DM associations are used to model the corresponding Senaps associations. Based on the Senaps conceptual model and its mapping to ProvONE and PROV-DM, we present SENProv - an ontology to capture and represent Senaps workflow provenance. Figure 4 highlights the most important classes and relationships that make up the SENProv ontology. The green ovals (i.e., PROV Entities), rectangles (i.e., PROV Activities), and pentagons (i.e., PROV Agents) represent the concepts in Senaps whereas yellow and blue presents PROV-DM and ProvONE concepts, respectively.

2.3 Provenance from Event Logs

Provenance captured from the event logs of Senaps, which are configured for INFO level logging. At INFO level, informational messages that are most useful are logged for monitoring and managing an application during execution. For example, an INFO level message describes the event type, the time, data used, and data generated by a workflow. Moreover, it considers an operator node and the model as a black box. Hence, INFO level logging enables the collection of coarse-grained provenance (Herschel et al., 2017).

An entry in a Senaps event log comprises of three main components: **DateTime**– Date and time of an event, **EventType**– the type of the event (e.g., EmptyWorkflowCreatedEvent, OperatorNodeAddedEvent, and DataUpdateEvent), and **Payload**– contains the information of the event including workflow and operator node execution ids, operator node, data nodes, ports, and data type (depending upon the event type). When a workflow executes in Senaps, the event log records twelve to fourteen different events for each operator node of the workflow. However, all the information required to capture provenance of an operator node execution is available from the payload of `'ExecutionRequestedEvent'` entry of the execution. Other event type entries of the operator node execution record incomplete and(or) duplicate information. Therefore, ProvAnalyser extracts the provenance from the payload of 'ExecutionRequestedEvent' and ignores other entries for the same operator node execution id while capturing provenance. The current implementation records the provenance of successfully executed workflows; however, in the future, we plan to capture unsuccessful workflow provenance to understand the root causes of workflow execution failure. This information is obtained from `'ExecutionSuccessfulEvent'` entry for an operator node execution of the workflow.

Provenance extraction from the log files is carried out by the **Log Parser and Filter** component of ProvAnalyser. The entries with event type 'ExecutionSuccessfulEvent' are filtered from the file, the

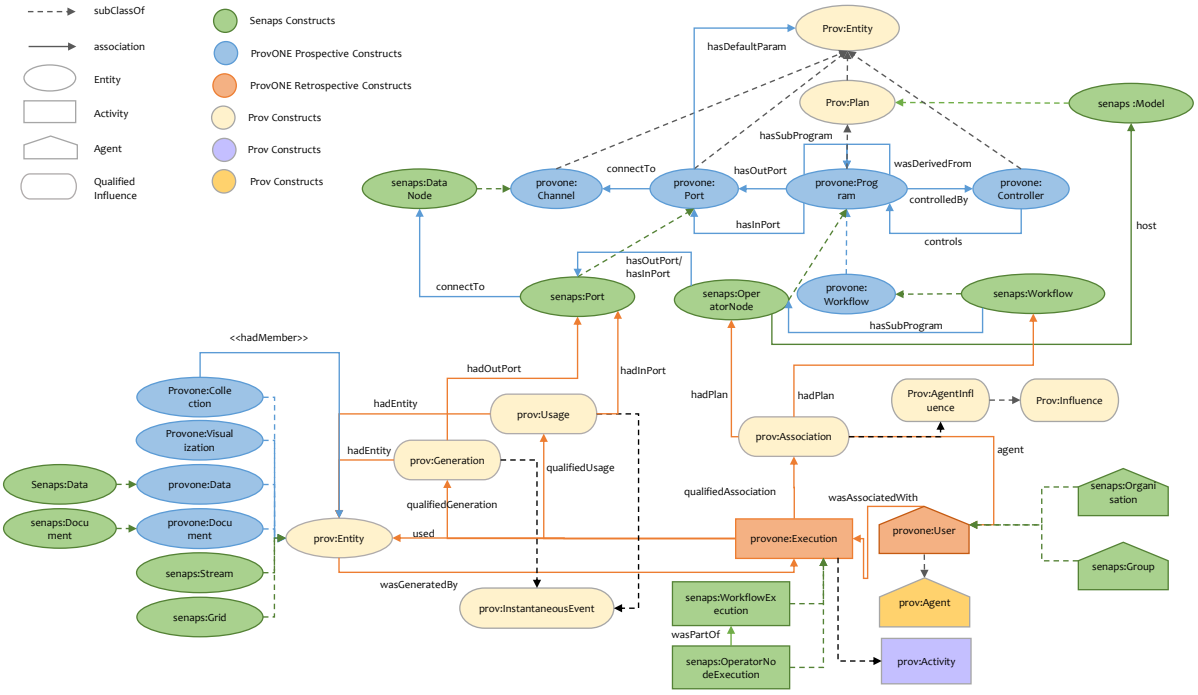

Figure 4: Core Structure of SENProv, showing relationship to PROV-DM and ProvONE - The constructs are represented in this diagram using PROV-like elements.

workflow execution Id for each such event is extracted from the payload and recorded into a `'Successful workflows list'`. Next, the entries with event type 'ExecutionRequestedEvent' are selected to retrieve the provenance of successfully executed workflows. The information about operator node, connected data nodes, model and ports are retrieved from the payload as raw provenance data, as shown in Listing 1. Using the SENProv, raw provenance data is transformed into structured provenance (i.e., an RDF document). ProvAnalyser retrieves prospective and retrospective provenance according to the SENProv model, as shown in Listing 2 and 3, respectively. The structured provenance is subsequently stored in the `Jena TDB`[7].

Implementation and Performance. The `Provenance Capturing module`, implemented in Java (jdk-1.8.0), processes the log files and uses Apache's Jena RDF API (apache-jena-3.7.0) to transform and store the structured provenance.

For the evaluation and testing purposes, we extracted provenance from the Senaps event log of 90 days. All the processing was performed on a 64-bit Windows 10 Enterprise computer using an Intel Core i7 6600U CPU with 2 cores and 8 GB memory. We processed log files of variable sizes (i.e., from 3 to 410 MBs), and the execution takes between 2 to 38 seconds to parse a log file, extract provenance from the file and store it in the RDF store.

However, the time ProvAnalyser takes to process a log file depends on the number of successfully executed workflows in the file and not on its size. Moreover, we collected provenance for 4658 workflow runs and 246,224 operator node executions in the RDF store of 2.29GB from log files of 6.29GB by

Listing 1: Raw Provenance.

```
{"workflowExecId" : "c49ff96d-cc5771b5d689",
"opNodeExecId" : "c49ff96d-forecast.template",
"opExecutionTime" : "2018-07-17T03:43:11.474Z",
"operatorNodeId" : "forecast.template-selector",
"modelId" : "apsim-template-selector",
"Ports" : [
    {
        "portId" : "location",
        "portDirection" : "Input",
            "connectedData" : {
                "dataNodeId" : "02b5ffde3e18",
                "dataNodeType" : "Document" }
    }
    {
        "portId" : "parameters",
        "portDirection" : "Input",
        "connectedData" : {
                "dataNodeId" : "7096195c361f",
                "dataNodeType" : "Document" }
    }
    {
        "portId" : "apsim_template",
        "portDirection" : "Output",
        "connectedData" : {
                "dataNodeId" : "673aeb335602",
                "dataNodeType" : "Document"}
    } ]
}
```

[7]https://jena.apache.org/documentation/tdb/

Listing 2: Prospective Provenance.

```
<c49ff96d-cc5771b5d689> a senaps:Workflow;

<forecast.template-selector> a senaps:OperatorNode;
   senaps:host <apsim-template-selector>;
   provone:hasInPort <location>;
   provone:hasInPort <parameters>;
   provone:hasOutPort <apsim_template>.

<apsim-template-selector> a senaps:Model.

<location> a senaps:Port;
   provone:connectTo [ a senaps:Document].

<parameters> a senaps:Port;
   provone:connectTo [ a senaps:Document].

<apsim_template> a senaps:Port;
   provone:connectTo [ a senaps:Document].
```

Listing 3: Retrospective Provenance.

```
<c49ff96d-cc5771b5d689> a senaps:WorkflowExecution;

<c49ff96d-forecast.template> a senaps:OperatorNodeExec;
   senaps:partOf <c49ff96d-cc5771b5d689>
   prov:atTime  2018-07-17T03:43:11;
   prov:qualifiedAssociation <c49ff96d-assoc-forecast>;
   prov:qualifiedUsage <c49ff96d-02b5ffde3e18-forecast>;
   prov:qualifiedUsage <c49ff96d-096195c361f-forecast>;
   prov:qualifiedGen <c49ff96d-673aeb335602-forecast>;
   prov:used <02b5ffde3e18>;
   prov:used <7096195c361f>.

<c49ff96d-assoc-forecast> a prov:Association;
   prov:hadPlan <forecast.template-selector>;
   prov:agent <Graincast>.

<c49ff96d-02b5ffde3e18-forecast> a prov:Usage;
   provone:hadInPort <location>;
   prov:hadEntity <02b5ffde3e18>.

<c49ff96d-096195c361f-forecast> a prov:Usage;
   provone:hadInPort <parameters>;
   prov:hadEntity <096195c361f>.

<c49ff96d-673aeb335602-forecast> a prov:Generation;
   provone:hadOutPort <apsim_template>;
   prov:hadEntity <673aeb335602>.

<673aeb335602> a senaps:Document;
   prov:wasGeneratedBy <c49ff96d-forecast.template> .

<02b5ffde3e18> a senaps:Document;

<7096195c361f> a senaps:Document;

<Graincast> a senaps:Group;
   prov:wasAssociatedWith <c49ff96d-forecast.template>.
```

using ProvAnalyser. This result of the provenance-enhanced RDF data being smaller in size than the raw logs echos other log-to-PROV experiences (Car et al., 2016).

3 QUERYING WORKFLOW PROVENANCE

ProvAnalyser can answer a wide range of relevant questions using the provenance ontology described in Section 2.2, including What actors (organisations or groups) were involved in executing a workflow? Which workflow was the most popular during a specific period? Find all the workflows which used a particular model. And list the parameters used in a particular workflow run.

Understanding a scientific workflow and reproducing its results are essential requirements to trust workflows and their results. These two requirements lead to the reuse of workflows and data generated by them across or within organisations. Therefore, our focus in this work is on use cases related to these two essential requirements. For instance, ProvAnalyser should be able to answer queries like **'track the lineage of the final output of a workflow'**. The lineage of output should explain which workflow generated it, when the output was generated, who is responsible for it, what dataset(s) and models are used while generating this output. How did the process use the input data, and how were the steps configured? The result of this query will enable a user to repeat a series of steps on original data to reproduce outcomes. This capability of a workflow engine is useful for both the clients and the developers of the workflow. A scientist needs provenance knowledge to assess the reliability of the outcomes or reuse a model in another workflow. Likewise, a workflow developer could be interested in investigating whether the workflow execution traces conform to the workflow structure by executing specific models in a particular order.

In this paper, we also discuss two additional use cases related to traceability and provenance analytics. This brings us to discuss four primary use cases for ProvAnalyser and provide their sample queries.

Use Case 1: Understandability– Explain a Workflow. This use case helps in understanding the workflow by producing the leading intermediate operators or models used in the execution of a particular workflow. A scientist could demand to examine workflow processes in detail to assess the reliability of results or to reuse operators in another workflow. A sample query is as follows:

What structure was followed by a given workflow execution trace? A typical understandability question to be addressed to understand the outcome of a complex scientific process. Listing 4 shows a SPARQL query to retrieve the structure of a workflow execution trace.

For a workflow execution, the query constructs the detail of a workflow structure. Consider an example of an execution of **Forecast Grains workflow** shown in Figure 1. For this execution, the result of the query identifies all intermediate operator nodes, their ports, and how the data was routed among the operator nodes as shown in Figure 5. Consequently,

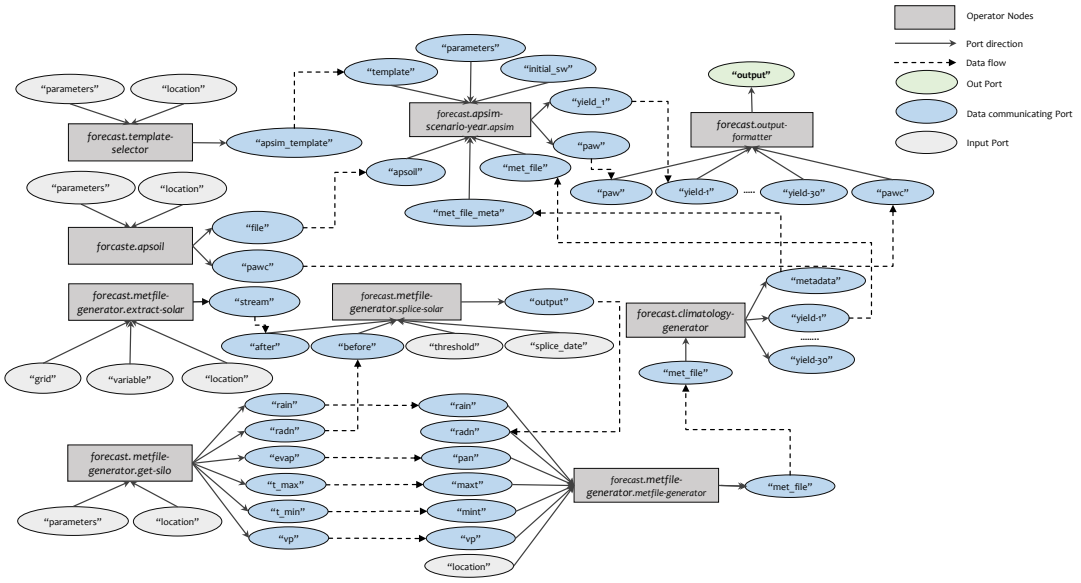

Figure 5: A graphical view of a SPARQL query result; the query is given in Listing 4.

Listing 4: SPARQL to get workflow specification.

```
PREFIX senaps:<http://www.csiro.au/ontologies/senaps#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX provone:<http://purl.dataone.org/provone#>
PREFIX prov:<http://www.w3.org/ns/prov#>

CONSTRUCT {
 ?sourceOpNode provone:hasOutPort ?outport.
 ?desOpNode provone:hasInport ?inport.
 ?sourceOpNode provone:controlledBy ?controllerURI.
 ?controllerURI provone:controls ?desOpNode.
 ?controllerURI rdf:type provone:Controller.}
 WHERE {
 <wfExecId> senaps:hasSubProgram  ?sourceOpNode.
 ?sourceOpNode senaps:operatorNodeId ?sourceOpNodeId;
        provone:hasOutPort ?outport.
 ?outport senaps:portId ?outportId.
 ?entityGen provone:hadOutPort ?outport;
        provone:hadEntity ?entity.
 ?entityUsed provone:hadEntity ?entity;
        provone:hadInPort ?inport.
 ?inport senaps:portId ?inportId.
 <workflowExecId> provone:hasSubProgram  ?desOpNode.
 ?desOpNode provone:hasInPort ?inport;
        senaps:operatorNodeId  ?desOpNodeId.
 BIND (URI(CONCAT(STR( ?sourceOpNode),\".\",
    STR(?outportId), \"_to_\", STR( ?desOpNodeId),
        \".\",STR(?inportId))) AS ?controllerURI)}
```

upon the query outcome, a user can comprehend the detailed structure of the workflow as shown in Figure 6.

Use Case 2: Reproducibility– Find Information to Reproduce. Organisations may want to reproduce their own or others' work. A scientist should be able to begin with, the same inputs and methods (models) used previously and observe if a prior result can be confirmed. This is a particular case of repeatability where a complete set of information is obtained to verify a final or intermediate result. In the process of repeating, and especially in reproducing, an output the scientist needs to know which models were used to derive an output and how the model used the input data. A sample query of the use case is:

Find what and how to use input data to result in a specific yield prediction. Listing 5 presents a SPARQL query to answer this question.

Listing 5: SPARQL to find input information.

```
PREFIX senaps:<http://www.csiro.au/ontologies/senaps#>
PREFIX provone:<http://purl.dataone.org/provone#>
PREFIX prov:<http://www.w3.org/ns/prov#>

SELECT DISTINCT ?model (?portId AS ?variableName) ?data
WHERE {
    <output> (prov:wasGeneratedBy/prov:used)* ?data.
    OPTIONAL {?data prov:wasGeneratedBy ?exec.}
    OPTIONAL {?usage provone:hadEntity ?data.
        ?usage provone:hadInPort ?port.
        ?port senaps:portId ?portId.
        ?opNode provone:hasInPort ?port.
        ?opNode senaps:host ?model. }
FILTER (!bound(?exec)) }
```

The query returns the details of the inputs to a workflow to generate a specific output, including input ids, ports the inputs were connecting to an operator node, and the model hosted by the operator node. For instance, for an output (outputId: `<42b838a7-786c-42a0-a4b9-f7dbed9df292>`) generated by an execution of **Forecast Grains workflow** the query returns all input ports in Figure 5, input data provided to these input ports, and models that used these input data.

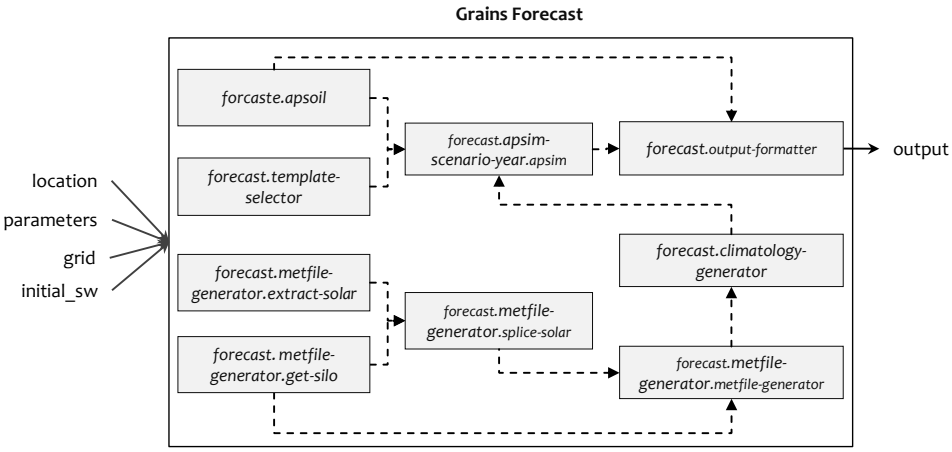

Figure 6: A detailed workflow for forecasting grains production on a location.

Use Case 3: Traceability– Trace the Effect of a Change. This use case traces the effect of a change. It identifies the scope of the change by determining workflows and their executions that are (or have been) affected. Moreover, tracing the effect can be used to minimise the re-computations to only those parts of a workflow that are involved in the processing of the changed data or model. A traceability related query is, **Identify all workflow executions that used (a specific version of) the APSIM model and group them by their organisations.** The result of this query helps to communicate all the organisations which are likely to be affected by a change in the APSIM model. Listing 6 shows the SPARQL syntax of this query.

Listing 6: SPARQL to trace the affect of change.

```
PREFIX senaps:<http://www.csiro.au/ontologies/senaps#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX provone:<http://purl.dataone.org/provone#>
PREFIX prov:<http://www.w3.org/ns/prov#>

SELECT DISTINCT ?orgs ?workflowExec
WHERE {
    ?opNodes senaps:host senaps:graincast.apsim.
    ?assoc prov:hadPlan ?opNodes;
        prov:agent ?orgs.
    ?orgs rdf:type senaps:Organisation.
    ?opExecution prov:qualifiedAssociation ?assoc;
            senaps:partOf ?workflowExec.
    } Group By ?orgs ?workflowExec
```

Use Case 4: Provenance Analytics. Provenance-based analytics help scientists to discover new research opportunities, identify new problems, and challenges hidden in the traces of workflow executions. Most importantly, it helps scientists discover and address anomalies. ProvAnalyser's current implementation can partially answer some provenance analytics related queries. For instance, a scientist may like to know:

Is the behavior in a second workflow execution conformant with the workflow's behavior in the first? This query helps impact (due to intentional changes in workflows) and(or) cause (due to accidental changes in workflows) analysis in case of any change in the behavior of workflow on two separate days. To date, ProvAnalyser can partially answer the query by providing the implicit workflow structure of two workflow executions using query presented in Listing 4.

4 DISCUSSION

ProvAnalyser achieved satisfactory performance in answering a range of relevant provenance queries and exhibits high usability compared to event logs. Nevertheless, some issues are planned to be addressed in the future.

4.1 Limitations

A significant concern is that ProvAnalyser currently captures retrospective provenance through an event log and infers partial prospective provenance using SENProv; it does not record exact prospective and workflow evolution provenance. Workflow specification and evolution provenance are required to address many provenance analytic queries. For instance, a scientist faces divergent outcomes during reproducibility analysis, i.e., two executions of the same workflow produce different results. The scientist is interested to know **what is (are) the reason(s) of divergent results of two executions of a workflow?** One such reason could be the data or workflow evolution, or it could be some unintentional changes in the

workflow. For ProvAnalyser to identify the cause, it should capture the workflow evolution and prospective provenance. Moreover, ProvAnalyser is capable of producing implicit prospective provenance through reverse engineering, as shown in Listing 4. However, for conformance checking (Moreau, 2015), a user is required to compare the implicit workflow specifications with explicit workflow specifications manually. The ProvAnalyser needs to capture workflow retrospective provenance explicitly to automate conformance checking.

Another limitation is the unavailability of a user-friendly provenance exploration and analysis mechanism. ProvAnalyser uses SPARQL as the only mechanism to query stored provenance. Although query-based access mechanisms (e.g., SPARQL, SQL, XPath or XQuery) are amongst the most popular provenance access methods (Oliveira et al., 2018b), it is usable for expert users (people with query language expertise) or, for naïve users, to answer pre-formulated queries. An appropriate method of provenance data visualisation or exploration can improve the data interpretation, facilitate decision making, and lead scientists to unexpected science discovery from the provenance traces.

4.2 Next Steps

Senaps users have well received ProvAnalyser, but its integration within the Senaps architecture requires some additional steps.

First, we intend to automatise the process of importing and processing the most recent event log. Currently, we update our provenance data weekly by importing new log files and extracting structured provenance from them. This solution has two limitations: it requires human intervention, and the system is updated only every week. We plan to fully automate the process of importing a new event log daily and extracting provenance from them. At a later stage, we intend to introduce hooks in the system to capture provenance directly from Senaps at the time when a workflow is submitted or executed instead of the event log.

Secondly, we plan to store provenance knowledge graph in PROMS (Car et al., 2014) that provides built-in features for provenance data validation and privacy. It allows detailed, system-specific, and less detailed system- independent descriptions provenance to validate through rulesets.

Finally, we plan to extend ProvAnalyser to address the limitations in Section 4.1 including capturing and storing workflow prospective & evolution provenance; and a visualization tool for interactively exploring provenance.

5 RELATED WORK

Workflow provenance has been studied in a variety of domains, including experimental science, business, and data analytics (Herschel et al., 2017). The motivation for employing workflow provenance in science is its ability to reproduce results from earlier runs, explain unexpected results, and prepare results for sharing and understanding. State-of-the-art scientific workflow engines Kepler (Altintas et al., 2006) Taverna (Oinn et al., 2004), WINGS/Pegasus (Kim et al., 2008), Galaxy (Goecks et al., 2010) and VisTrails (Bavoil et al., 2005) automatically capture workflow provenance in the form of execution traces. Moreover, there exist stand-alone approaches for provenance capturing and analytics (Oliveira et al., 2018b). However, most of the solutions often rely on proprietary formats that make interchanging provenance information difficult. Furthermore, these systems and approaches harvest provenance directly from the system at runtime workflow execution traces rather than log files, which requires systems' source code instrumentation.

For employing log files to understand the root causes of failures, LogMaster (Fu et al., 2012) uses system logs for extracting event correlations to build failure correlation graphs. SherLog (Yuan et al., 2010) leverages large system logs to analyse source code of the system. Jiaang et al. (Jiang et al., 2009) proposes a mechanism for root cause analysis of failure in large systems by combining failure messages with event messages. Xu et al.(Xu et al., 2009) detect problems in large scale systems by mining logs combined with the source code that generated the logs. Gaaloul et al. (Gaaloul et al., 2009) analyse workflow logs to discover workflow transaction behaviors and to improve and correct related recovery mechanism subsequently. Likewise, NetLogger (Gunter et al., 2000) collects and analyses event logs for the performance of distributed applications, but it needs source code instrumentation. However, all these systems do not explicitly collect provenance information from log files. Although logs contain pertinent information for error analysis, they can also be employed for garnering the relevant information about workflows execution and data objects.

While most previous log analysis has been done to understand the root causes of failures, little work exists on extracting workflow provenance information from log files. Car et al. (Car et al., 2016) extracted PROV-O compliant provenance from Web service log

to generate web service request citation. Ghoshal and Plale (Ghoshal and Plale, 2013) presented the most relevant approach to ProvAnalyser. They explore the options of deriving workflow provenance from existing log files. However, their focus is on collecting provenance from different types of logs of distributed applications. Our approach leverages Senaps event log to capture interoperable provenance and analyse it to understand and reproduce workflow outputs.

6 CONCLUSION

This work shows that provenance data can be captured from scientific workflow systems' event logs that can verify the quality of their data products and allow the analysis of workflows execution traces to make them understandable and reusable. The logs can be filtered and transformed into standardised provenance data using a specialised model. This transformation allows the recording of valuable information into a standardised and workflow system-independent format that is both interoperable and intelligible to the provenance users. Also, the storage volumes of the provenance required to perform data and workflow quality assessments and analysis are smaller than the log size, indicating the practical scalability of this transformation process. While the workflow execution provenance recorded from the event log can answer most of the user queries, it is not always enough and, where it is not, workflow prospective provenance can be inferred and used. However, to enable comprehensive provenance analytics, the systems should consider capturing prospective and evolution provenance information in their logs.

REFERENCES

Altintas, I., Barney, O., and Jaeger-Frank, E. (2006). Provenance collection support in the kepler scientific workflow system. In *Provenance and Annotation of Data*, pages 118–132, Berlin, Heidelberg. Springer.

Bavoil, L., Callahan, S. P., Crossno, P. J., Freire, J., Scheidegger, C. E., Silva, C. T., and Vo, H. T. (2005). Vistrails: enabling interactive multiple-view visualizations. In *VIS 05 IEEE Visualization*, pages 135–142.

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.

Car, N. J., Stanford, L. S., and Sedgmen, A. (2016). Enabling web service request citation by provenance information. In *Provenance and Annotation of Data and Processes - 6th International Provenance and Annotation Workshop, McLean, VA, USA, June 7-8, 2016, Proceedings*, pages 122–133.

Car, N. J., Stenson, M. P., and Hartcher, M. (2014). A provenance methodology and architecture for scientific projects containing automated and manual processes. [accessed through: http://academicworks.cuny.edu/cc_conf_hic/57].

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 12-Dec-2019].

Curcin, V. (2017). Embedding data provenance into the learning health system to facilitate reproducible research. *Learning Health Systems*, 1(2):e10019.

Fu, X., Ren, R., Zhan, J., Zhou, W., Jia, Z., and Lu, G. (2012). Logmaster: Mining event correlations in logs of large-scale cluster systems. In *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pages 71–80.

Gaaloul, W., Gaaloul, K., Bhiri, S., Haller, A., and Hauswirth, M. (2009). Log-based transactional workflow mining. *Distributed and Parallel Databases*, 25(3):193–240.

Garijo, D. and 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, pages 47–56, New York, NY, USA. ACM.

Ghoshal, D. and Plale, B. (2013). Provenance from log files: A bigdata problem. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 290–297, New York, NY, USA. ACM.

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

Gunter, D., Tierney, B., Crowley, B., Holding, M., and Lee, J. (2000). Netlogger: A toolkit for distributed system performance analysis. In *Proceedings 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (Cat. No. PR00728)*, pages 267–273. IEEE.

Herschel, M., Diestelkàmper, R., and 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.

Jiang, W., Hu, C., Pasupathy, S., Kanevsky, A., Li, Z., and Zhou, Y. (2009). Understanding customer problem troubleshooting from storage system logs. In *Proccedings of the 7th Conference on File and Storage Technologies*, FAST '09, pages 43–56, Berkeley, CA, USA. USENIX Association.

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

Moreau and Missier (2013). World Wide Web Consortium "PROV-DM: The PROV Data Model" W3C Recom-

mendation . https://www.w3.org/TR/prov-dm/. [Online; accessed 12-Dec-2019].

Moreau, L. (2015). Aggregation by provenance types: A technique for summarising provenance graphs. *arXiv preprint arXiv:1504.02616*.

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

Oliner, A. and Stearley, J. (2007). What supercomputers say: A study of five system logs. In *37th Annual IEEE/IFIP Int'l Conf on Dependable Systems and Networks*, pages 575–584. IEEE.

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

Oliveira, W., Oliveira, D. D., and Braganholo, V. (2018b). Provenance analytics for workflow-based computational experiments: A survey. *ACM Comput. Surv.*, 51(3):53:1–53:25.

Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 117–132, New York, NY, USA. ACM.

Yuan, D., Mai, H., Xiong, W., Tan, L., Zhou, Y., and Pasupathy, S. (2010). Sherlog: Error diagnosis by connecting clues from run-time logs. *SIGPLAN Not.*, 45(3):143–154.