Linking Abstract Plans of Scientific Experiments to their Corresponding Execution Traces

Milan Markovic  
University of Aberdeen  
Aberdeen, UK  
milan.markovic@abdn.ac.uk

Daniel Garijo  
Information Sciences Institute,  
University of Southern California  
Los Angeles, USA  
dgarijo@isi.edu

Peter Edwards  
University of Aberdeen  
Aberdeen, UK  
p.edwards@abdn.ac.uk

ABSTRACT

Provenance describes the creation, manipulation and delivery processes of scientific results; and has become a crucial requirement for debugging, understanding, inspecting and reproducing the outcomes of scientific publications. Scientific experiments, in particular computational workflows, often include provenance collection mechanisms that link execution traces to their respective planned specifications. Such provenance traces are typically very fine-grained, and may quickly become too complex or difficult for humans to interpret. In this paper we describe our approach to represent workflow plans and provenance at different levels of abstraction. We describe EP-Plan, a W3C PROV ontology extension and we illustrate our approach with a use case using the WINGS workflow system.

KEYWORDS

Plan, scientific workflows, provenance, abstractions

1 INTRODUCTION

Scientific workflows describe the computational steps and data dependencies that are necessary to carry out a scientific experiment [13]. Scientific workflows can be found in a wide range of domains, ranging from Geosciences to Bioinformatics, as they have demonstrated their utility for reproducing previous experiments, improving standardization practices in a research lab and educating students on existing methods [2]. Scientific workflow systems usually have the ability to capture the provenance traces of executed experiments, to support inspection of results and debugging of workflow errors [12]. The W3C recommendation PROV-O [9] is a standard model for representing provenance of any entity in the Web, by exposing the series of activities that used or generated such entities. PROV-O is often used as a reference model by workflow engines. Similarly, scientific workflows may contain high-level abstract steps that lead to different implementations depending on the algorithms selected for execution [4]. Linking these abstract plans with their execution traces requires additional mechanisms which have not been defined in P-Plan or other recent efforts for provenance representation in scientific workflows such as the Research Object Model [1] and ProvOne\(^1\) specifications.

In this paper, we use the Extended P-Plan ontology (EP-Plan)\(^2\) to link together different abstractions of scientific workflow plans and their execution traces described using PROV-O.

We first detail the challenges for linking provenance to abstract plans in Section 2 by using examples from the WINGS workflow system [5]. We then describe how we have addressed these challenges with the EP-Plan ontology in Section 3, and we conclude with a discussion of our future work.

2 ABSTRACT PLANS AND PROVENANCE IN SCIENTIFIC WORKFLOWS

During their lifecycle, scientific workflows may be defined at different levels of abstraction, from an abstract original specification by a user to a fully detailed execution plan prepared for a workflow engine [4]. Here we focus on supporting three common use cases in scientific workflows:

- *Collections of activities and entities*: Plans may contain abstractions that summarize execution activities to be performed in parallel. Figure 1 shows an example using a workflow for water quality analysis in the WINGS workflow system [6]. As shown on the left of the figure, some steps represent collections of executions, depicted as stacked boxes. For example, the step *MetabolismCalcEmpirical* receives a collection of *HourlyData* files which will be executed in parallel. The right side of the figure shows a fragment of the corresponding execution plan of the workflow on the left, after a user has specified the input files and hence the system has prepared the full execution plan. Since provenance is tracked at the granularity of the execution plan (shown at the right of the figure), it is necessary to define properties to group entities and associate them to the corresponding abstract plan.

- *Workflow fragments*: Workflow systems often include the ability to define sub-workflows to simplify complex workflow plans. A sub-workflow would then appear as a single step in the bigger workflow. While this mechanism is helpful

\(^1\)http://purl.org/provone
\(^2\)https://w3id.org/ep-plan
for easing the understandability of the scientific workflow, it requires the means to link the provenance for the sub-workflow execution back to the provenance of the workflow where it was included.

- **Execution summaries**: When workflow execution plans become complicated, the corresponding provenance traces may be too convoluted to explore by users who only want to know more about the inputs used to generate the result of a workflow. Figure 2 shows a simple example of this behaviour: the workflow execution summary on the left represents the execution of the workflow as a single activity. The provenance trace on the right represents the full workflow execution trace. Both the execution summary and the full provenance trace represent valid views of a workflow execution, and should be linked together.

Linking these different levels of granularity together is crucial for workflow systems to inform a user in case of execution errors. To further support users’ ability to understand errors and how plans and their fragments may be reused, plans should also contain additional metadata that provide information about the context in which the individual planned steps were deployed and executed. This may include information about any associated constraints (e.g. for input validation), the agents expected to perform individual steps, objectives the plan is trying to achieve, etc.

While specifications such as the Research Object Model, D-PROV [11] ProvOne or CWL-PROV [7] define mechanisms to describe sub-workflows and entity collections (e.g., by defining part of relationships) they do not define clear mechanisms to link together provenance traces at different levels of granularity. Below we describe how we address these issues with the EP-Plan ontology.

### 3 USING EP-PLAN TO REPRESENT PLANS AT DIFFERENT LEVELS OF ABSTRACTION

EP-Plan builds on P-Plan[^3][3], a vocabulary designed for aligning simple plans to their corresponding provenance traces. EP-Plan was designed for cross domain applications (e.g. the use of EP-Plan for enhancing Internet of Things deployments is detailed in [10]) and uses `ep-plan:Step` to denote any planned process, and `ep-plan:Variable` to represent inputs and outputs of steps.

[^3]: p-plan namespace: http://purl.org/net/p-plan
Figure 3: An overview of a subset of EP-Plan concepts for describing and linking plan specifications with their execution traces.

Figure 3 illustrates a subset of EP-Plan concepts that define mechanisms for linking plan specification and execution traces at different levels of abstraction. Both steps and variables belong to ep-plan:Plan (modelled as a subclass of prov:Plan defined in PROV-O) and are linked to their corresponding executions described as ep-plan:Activity and ep-plan:Entity (modelled as subclasses of prov:Activity and prov:Entity). A workflow execution typically produces an execution trace that consists of a number of activities and entities representing instantiations of different parts of a plan. In EP-Plan, a single execution trace is grouped by ep-plan:ExecutionTraceBundle (a subclass of prov:Bundle). A single plan specification may then be linked to multiple execution traces using prov:wasDerivedFrom. To allow linking of different levels of workflow abstractions, EP-Plan provides mechanisms to group related workflow steps defined at a finer level of detail together as a sub-plan that then further describes a step of a more abstract plan denoted as ep-plan:MultiStep. The left side of Figure 4 illustrates a high level abstraction of a workflow plan (:SummarizedWf) containing a single ep-plan:MultiStep (:ExecuteWorkflowStep) that is then described in more detail on the right side of the figure as a sub-plan (:ExecutedWf). In the same figure, the abstract workflow (:SummarizedWf) also includes abstractions of two variables (:InputFilesVar and :OutputFilesVar) described using the class ep-plan:MultiVariable. In the sub-plan specification, each of the multivariables is decomposed into two individual variables (e.g., :File1Var and :File2Var) and linked using ep-plan:hasPart.

Figure 5 illustrates an example execution trace with two execution trace bundles corresponding to the plan and its sub-plan shown in Figure 4. Execution trace elements corresponding to multi variables defined in the :SummarizedWf plan (see Figure 4) correspond to trace elements of the type ep-plan:EntityCollection.

\[^4\text{prov namespace: http://www.w3.org/ns/prov#}\]
\[^5\text{Links ep-plan:correspondsToVariable that link ep-plan:EntityCollection from the execution trace record to ep-plan:MultiVariable in the plan specification are not shown in the figure.}\]

Figure 4: An example illustrating decomposition of ep-plan:MultiStep into a sub-plan and linking of variables across different levels of plan abstractions.

Figure 5: An example description of execution traces corresponding to workflows defined at different levels of abstraction.
which is a subclass of prov:Collection (see :InputFiles and :OutputFiles in Figure 5). The usage and generation of these entity collections is ascribed to a trace element :WorkflowExecution (modelled as ep-plan:MultiActivity) using relationships prov:used and prov:wasGeneratedBy. The trace element :WorkflowExecution corresponds\(^6\) to the plan element executeWorkflowStep shown in Figure 4. The right side of Figure 5 shows a more detailed execution trace corresponding\(^7\) to the :ExecutedWf plan specification shown on the right side of Figure 4. Instantiations of plan variables are captured as instances of ep-plan:Entity (e.g. see :File1) and instantiations of steps are captured as instances of ep-plan:Activity (e.g. see :Aggregate). Relationships prov:hadMember are used to link trace elements corresponding to abstract multivariables (modelled as ep-plan:EntityCollection in :AbstractExecutionTrace) and their more detailed description in :ExecutionTrace produced by the sub-workflow specification.

To summarise, using the mechanisms outlined above, EP-Plan enables modelling of abstract workflow specifications by collapsing multiple steps and variables into aggregated plan elements (i.e. multisteps and multivariables). Sub-plans containing more detailed descriptions of plan abstractions may be linked and reused by different plans (i.e. as workflow fragments), as these are modelled as individual plan specifications (including any relevant metadata). Furthermore, by leveraging the concept of collections, we are also able to maintain links between different abstractions of execution traces without violating PROV-O semantics.

Finally, in contrast with P-Plan, EP-Plan provides a richer vocabulary for capturing plan metadata which (for reasons of space) is not discussed in detail in this paper. Briefly, this includes the ability to associate descriptions of agents that are allowed to execute different steps of a plan, to link descriptions of policies, and to describe specifications of how data should be exchanged between steps. Plan elements can be also associated with descriptions of constraints that provide a high level reference to any restrictions that can be linked to and evaluated against elements of an execution trace. EP-Plan also enables descriptions of objectives to be associated with the plan. Objectives may then be linked to the individual plan elements that achieve them. Each element may also be linked to a rationale (e.g. user-readable description) which details why the element was included in the plan specification. These concepts are important for describing the execution context of a scientific experiment. This may include, for example, specifications of individual scientists that are allowed to control certain steps of a plan, links to a data protection policy applicable to an experiment using sensitive or personal data, constraint descriptions which provide further information about the portions of a workflow that failed to execute due to constraint violation, etc.

### 4 CONCLUSIONS & FUTURE WORK

In this paper, we introduced the EP-Plan ontology for describing scientific experiments. In particular, we focused on describing experiments at different levels of abstraction. In our future work we aim to focus on using EP-Plan to enhance the provenance traces generated by WINGS (which currently uses the OPMW ontology\(^8\)) with additional plan descriptions. OPMW extends both P-Plan and Prov-O and therefore it should be possible to align existing provenance descriptions generated by WINGS with the EP-Plan vocabulary. The WINGS system also uses semantic implementations of constraints to plan and execute scientific workflows [8]. We will explore how these can be mapped to the constraint concepts defined in EP-Plan and hence included as apart of the experiment metadata with the plan specification.

### ACKNOWLEDGMENTS

The work described in this paper was funded by the award made by the RCUK Digital Economy programme to the University of Aberdeen (EP/N028074/1), a SICSA FECE travel award, the Defense Advanced Research Projects Agency with award W911NF-18-1-0027, the SIMPLEX program with award W911NF-15-1-0555 and from the National Institutes of Health under awards 1U01CA196387 and 1R01GM117097.

### REFERENCES


---

\(^{6}\)Links ep-plan:correspondsToStep that link ep-plan:MultiActivity from the execution trace record to ep-plan:MultiStep in the plan specification are not shown in the figure.

\(^{7}\)Links ep-plan:correspondsToVariable and ep-plan:correspondsToStep that link ep-plan:Entity and ep-plan:Activity from the execution trace record to ep-plan:Variable and ep-plan:Step in the plan specification respectively are not shown in the figure.

\(^{8}\)http://www.opmw.org/model/OPMW/