

# Exploiting provenance to make sense of automated decisions in scientific workflows

Paolo Missier, Suzanne Embury, and Richard Stapenhurst

School of Computer Science, University of Manchester  
{pmissier,suzanne,stapenr5}@cs.man.ac.uk

**Abstract.** Scientific workflows may include automated decision steps, for instance to accept/reject certain data products during the course of an *in silico* experiment, based on an assessment of their quality. The trustworthiness of these workflows can be enhanced by providing the users with a trace and explanation of the outcome of these decisions. In this paper we present a provenance model that is designed specifically to support this task. The model applies to a particular type of sub-workflow that is compiled automatically from a high-level specification of user-defined, quality-based data acceptance criteria. The keys to the effectiveness of the approach are that (i) these sub-workflows follow a predictable pattern structure, (ii) the purpose of their component services is defined using an ontology of Information Quality concepts, and (iii) the conceptual model for provenance is consistent with the ontology structure.

## 1 Introduction

Modern experimental science is increasingly data-intensive: a typical *in silico* experiment involves the coordinated execution of a number of processes that produce, consume, transform and analyse data. Automating these processes bears the promise of increasing the rate at which new scientific results can be produced. At the same time, however, scientists are also responsible for making sure that the data produced by these experiments is sound and scientifically of good quality. Let us mention two of the factors that may contribute to the production of invalid output from an e-science experiment. The first is the increasing reliance on public data and service resources that are contributed by multiple parties within a scientific community, who normally do not offer guarantees of data quality control (or service accuracy). Because of this, low quality in the input may be expected. And secondly, errors can be introduced due to the inherent complexity and variability of the scientific experiments that produce the data. Some of these problems have been surveyed and classified for the case of transcriptomics and proteomics data, for example [6]. When these errors go undetected, because of a lack of appropriate quality controls either by the experimenter, or by the data provider, user scientists face the risk of inadvertently using using poor data that may invalidate the conclusions drawn from their own experiments.

In this paper we argue that, when the experimental process is implemented as a workflow, the analysis of provenance trails collected from workflow executions can play an important role in supporting the experimenters' claim of their results' soundness. By the term *workflow provenance* we mean *metadata that is collected during the execution of the workflow, in order to enable various types of post-mortem analyses on its outcome*. In particular, we are going to exploit provenance metadata to explain and justify the quality-based decisions made by a completely automated workflow on the user's behalf, in particular regarding which data elements are deemed acceptable based on their quality estimates. This problem is complicated by the potentially arbitrary nature of the processors that compose the workflow, as well as of the workflow structure. As has been noted [5], black-box processors that are not further annotated limit the ability to use the provenance log for explanation purposes, and similarly, an arbitrary workflow structure imposes a generic presentation model.

We do not propose a general solution to this problem. Instead, our approach is focused on a specific type of quality-based decision processes, and stems from three key design principles. Firstly, we take the stance that quality assurance in the workflow context is described by a process in its own right, which can be deployed as a part of the workflow itself, or as a sub-workflow. We call such process a *quality workflow*. Secondly, quality workflows are automatically generated from higher-level specifications, in a model-driven fashion, making their structure and their composing services *predictable*. And finally, the services that compose a quality workflow are described as part of an ontology of Information Quality concepts. As we will see in Section 3, this allows us to create a data model for provenance that follows the structure of the ontology. This uniformity of representation has at least two advantages. Firstly, we can query the provenance model using the ontology as a schema; and secondly, we can describe the relationships among elements in the provenance model in terms of semantic properties among their corresponding classes.

The combination of these three design principles make it possible to exploit provenance to provide users with a high-level, "semantic" view of quality-based decisions, in a way that would not be possible when dealing with arbitrary workflows.

Based on these premises, we present a detailed provenance model that is specifically dedicated to analysing quality workflows. We view this as only one specific case of an otherwise general confluence between model-driven workflow design, semantic annotation of services, and provenance modelling. Note that the examples used in the paper are set in the context of e-science workflows; also, the implementation of the provenance model described in the paper uses the Taverna workflow language [10, 7], part of the myGrid suite of middleware tools for e-science<sup>1</sup>. Neither of these is a limitation, however: the notion of quality workflows is completely general and applicable to other domains, and the provenance model does not contain any Taverna-specific element.

---

<sup>1</sup> <http://mygrid.org.uk>

The idea of semantic provenance models that are tailored to special-purpose workflows is not common in the literature, although various mature systems provide interesting ways to visualize provenance, i.e., VisTrail [3]. In fact, the ability to query provenance information at different levels of abstraction is listed as one of the many *desiderata* for provenance systems by Chapman and Jagadish [4] (it is listed as number IX).

The recent work on the *Zoom* provenance query prototype by Biton *et al.* [2] is relevant, in that it advocates a tailoring of provenance views to the needs of specific users, rather than just giving access to the enormous bulk of the raw logs in all their detail. We see their work as complementary to our own. We allow users to ignore the detail of the quality assessment aspects of the workflow (which could be packaged up into a “composite module”, to use the terminology of Biton *et al.*) until query time. At this point, the provenance browser provides an abstract view over provenance, that is based not on a user-specified view, but on the high-level model from which the quality sub-workflow was produced. Clearly, a number of abstract views of workflows could be supported by adopting mechanisms similar to those suggested in [2].

## 2 Quality-based decision processes

The quality assurance problems that motivate our work could, in principle, be alleviated by convincing data and service providers to deploy their own quality assurance procedures during data generation, maintenance, and provisioning. Besides being impractical, however, this proposition assumes that standardised quality estimation procedures can be developed. This, however, often contrasts with the very nature of interesting e-science data, which results from cutting-edge research conducted using new and experimental techniques that tend to change rapidly, not lending themselves well to standardisation.

Even when the data provider makes appropriate, objective quality metrics available, the user scientists are still faced with a decision problem, namely whether to accept or reject certain data based on its quality characteristics. Although the determination of data acceptability is based on objective metrics, the user’s perception of whether the data is fit for use, given its quality characteristics, also plays a part: some types of error, or approximation, can be tolerable for some types of applications, but not for others, and different users may attach different importance to quality.

### 2.1 Example

To make these considerations concrete, consider a real-life case study in the domain of qualitative proteomics [1], i.e., concerning the identification and functional characterization of proteins from a cell sample. The experiment includes an *in vitro* portion whereby a mass spectrometer is used to quantify the peptide masses in the sample, followed by an *in silico* portion where the observed masses are matched against theoretically computed masses for a large collection

of known proteins. The critical step in the latter portion of the experiment, denoted **Identify Proteins** in the Taverna workflow fragment of Fig. 1, is the invocation of the matching service. We can view this service as a dedicated search engine that operates on sequences of peptide masses. The results of the search invariably include partial matches as well as exact matches. Although some of these matches may turn out to be false positives, the experimenter has not simple way to make that determination. Ideally, a quality-based data acceptance criteria would be able to accept/reject individual matches based on their likelihood of being a false positive.

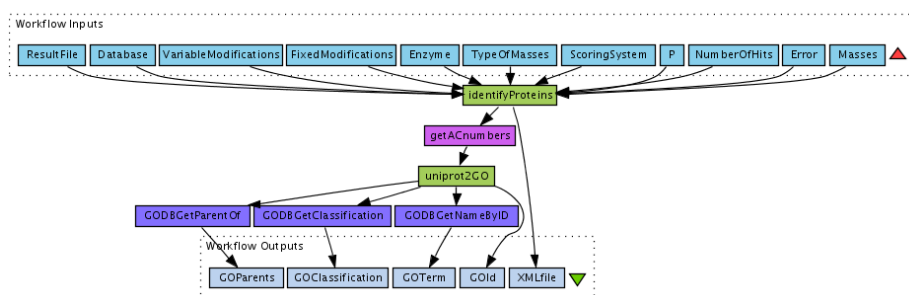


Fig. 1: A simple proteomics workflow with a potentially imprecise search processor

To help determine the reliability of each reported match, implementations of this service (**Imprint** is the homegrown service used in our example) typically do provide additional metadata along with the match, including for example the *Hit Ratio*, i.e., the number of peptide masses matched, divided by the number of peptide masses submitted to the search (additional metadata that is required in this example is omitted for simplicity). Recent research [11] has shown a strong correlation between a simple score model for matches based on this and other readily available indicators, and the likelihood of false positives. Using this predictive score model to rank the output of the matching service, in combination with a user-defined threshold, experimenters have an effective way to make their quality acceptance criteria formal and automatically computable.

## 2.2 Structure of the decision process

This example highlights the main elements of a quality-based decision process that is applied to a dataset, in this case a collection of protein matches: first a set of objective metadata elements, i.e., the *Hit Ratio*, is collected in order to compute a predictive quality model (the match score). We will refer to the metadata elements as *quality evidence*, and to the quality model as *quality assertion*. Then, a threshold is applied in order to partition the ranked protein matches into the two classes “accept” and “reject” –this is an example of a *quality condition*.

Note how the process combines purely objective elements, the evidence, with a predictive model, the assertion, and with a subjective element, the threshold. Generalising from the example, in [9] and [8] we have formally described a broad class of quality processes that take an input dataset and compute a partition of the dataset into *quality classes*, such as “accept” and “reject”. These processes share the structure just described, namely they:

- collect quality evidence from the data and the surrounding operating environment. In the example, the required evidence is either supplied by the search engine, or can be derived independently from its output;
- compute one or more quality assertions, using the collected evidence as input;
- evaluate a quality condition that assigns one quality class to each element in the input dataset, based on the values of the quality assertions.

In [9] we have coined the term *Quality View* to denote a formal specification of such a quality process. The process is abstract in that it does not include any indications regarding its implementation. A Quality View specifies three types of function, one for each of the steps listed above, namely (i) annotation functions that associate quality evidence metadata to the input dataset; (ii) quality assertion functions that associate quality assertions to the data based on the evidence, and (iii) quality actions that compute a quality classification based on the assertions.

### 2.3 Compiling quality processes to workflows

Quality Views are defined as part of a workbench for Information Quality management, called *Qurator* [9]. Using Qurator, e-scientists may define their own quality metrics for specific types of data as quality assertion functions, and then specify Quality Views in order to apply those metrics to the data. Quality Views are most useful when they are deployed as filters within larger, user-defined processes. For this reason, Qurator includes a compiler that translates Quality Views into workflows, specifically targeted at the Taverna workflow system. An example of such a *quality workflow*, designed to work with the example protein identification workflow of Fig. 1, is shown in Fig. 2.

The compiler assumes that all the annotation and assertion functions that are part of the Quality View have been implemented as Web Services. With this assumption, those functions translate simply to Taverna processors that perform service invocations. Specifically, the example workflow includes one annotation processor, `InprintOutputAnnotator`, and three quality assertion processors, for instance `PIScoreClassifier`. The action step at the end evaluates an expression on the values of any of the assertions computed by these processors, for instance “`HitRatio > 0.67 and PIScoreClassifier = 'high' and ...`”. Data elements that do not satisfy the condition are placed in the “reject” output of the processor, which is typically not connected to any other processor. Thus, this mechanism can be used in particular to filter out protein identifiers that rank too low, when a user-defined threshold is used in combination with the score model described in Section 2.1.

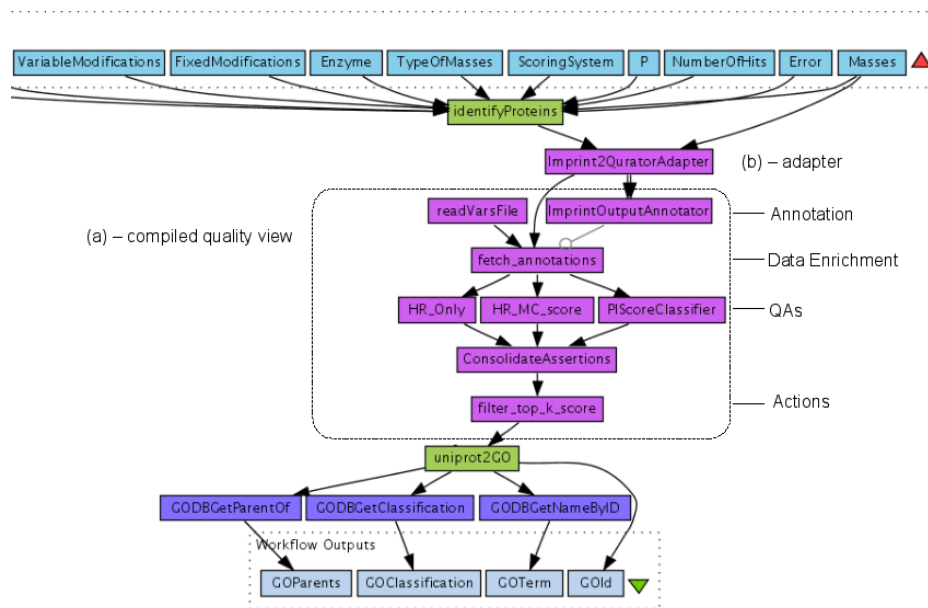


Fig. 2: A quality workflow deployed as part of the proteomics experiment

## 2.4 Role of provenance

As this example shows, quality workflows may have an impact on the outcome of a workflow, for instance by removing the likely false positives. It is therefore important for users to understand the effects of the quality workflow on the output of its original workflow. For this, the Quarator workbench includes a provenance component that is specialised to operate exclusively on quality workflows, providing users with a high-level trail to explain how a certain decision was reached. Specifically, the component supports the following tasks, among others:

- visualize the partitioning of the input data set into quality classes, as defined upon evaluating the action condition;
- for each data element, visualize the entire trail of transformations that contributed to its quality classification. This includes quality assertion values that were used in evaluating the condition, the names and types of the quality assertion functions, the values for their input quality evidence, and the annotation functions used to compute the evidence;
- visualize the different quality classification outcomes obtained over a series of workflow executions, highlighting the differences among the quality workflow settings (e.g. the action condition). Is a certain data element consistently rejected or accepted, for example? or is its acceptance particularly sensitive to a threshold configuration?



of a Quality View (data, quality metadata, and processors) are assigned a *semantic type*, i.e., a reference to a class in an ontology of Information Quality (IQ) concepts. To illustrate, consider Fig. 4, where the main concepts are shown along with their properties<sup>2</sup>. The leftmost part of the figure shows some of the

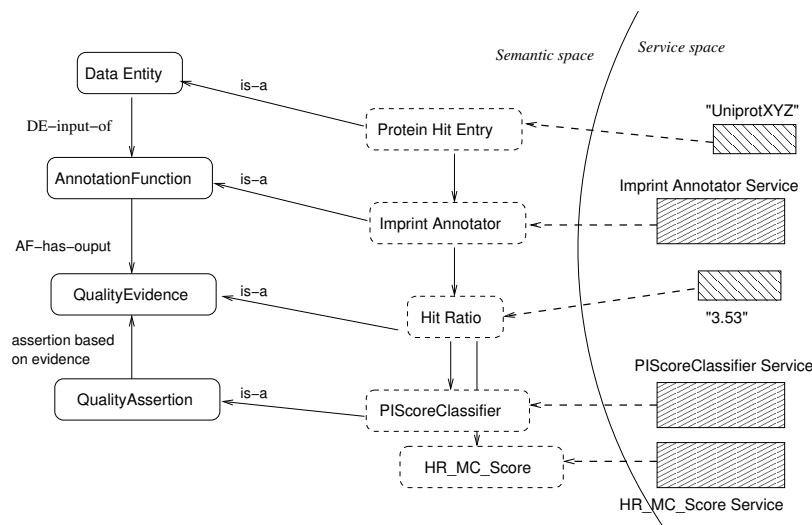


Fig. 4: A fragment of the Information Quality ontology with semantic typing of services

domain-independent ontology classes: an **Annotation function** computes values that are instances of **Quality Evidence** classes, using instances of **Data Entity** as input. Similarly, a generic **Quality Assertion** is based upon **Quality Evidence** values. On its right are specializations of each of these generic classes to domain-specific sub-classes, in this case associated to protein identification data (**Imprint** is the name of the specific protein matching tool used in the main workflow). Together, these classes and properties belong to a “semantic space” of symbols that can be used to give meaning to objects in the “service and data space”, on the right in the figure. In this space we find specific implementations of the functions as Web Services, as well as actual values for data and metadata elements (the patterned boxes).

When a Quality View is compiled into a quality workflow, objects in the service and data space are annotated with references to classes in the semantic space. As we will see shortly, by creating a data model for the provenance component that follows the structure of the ontology we are able to view provenance metadata as instances of the ontology classes. This uniformity of representation has at least two advantages. Firstly, we can query the provenance model using the ontology as a schema; and secondly, we can describe the relationships among

<sup>2</sup> This is a vastly simplified fragment of the ontology. For a complete account see [8].

elements in the provenance model in terms of semantic properties among their corresponding classes, leading to a presentation model for provenance that is close to the scientist’s intuition of the intended workflow behaviour.

Since the IQ ontology is specified using the OWL Semantic Web language<sup>3</sup>, these design decisions lead naturally to RDF<sup>4</sup> as the data model of choice for provenance. This is in accordance with common Semantic Web practice, whereby we can assign a semantic type to arbitrary RDF resources (by means of the pre-defined RDF(S) property `rdf:type`). In particular, some of the instances of the provenance schema have a semantic type that corresponds to the ontology classes shown in Fig. 4.

The provenance model consists of two parts. The first part, called the *static model*, is an RDF graph that describes elements of a quality workflow, as they are specified at compilation time, while the *dynamic model* is populated with actual provenance data during each workflow execution.

### 3.2 Static model

A fragment of the static model for our running example is shown in Fig. 5. This RDF graph contains two resources, namely the two nodes on the left in the figure. The ovals in the graph are RDF resources, identified using a unique URI, while the square boxes are *literals*, i.e., constant values, and directed arcs denote binary properties between any two nodes, the *subject* and the *object* of the property. Nodes can be *anonymous* (also called *b-nodes*), i.e., their URI is internal and system-defined rather than user-defined, and thus it is not shown.

The first of the two nodes on the left represents the only quality action in the workflow, i.e., `filter_action`, and it carries the definition of the action expression that is used to identify the “accept” data elements. The second is the root of a sub-graph that represents one of the quality assertion functions, along with its input and output variables, having semantic type `PIScoreClassifier`. Similarly, each input variable is an RDF resource too, consisting of a name (the literal) and a semantic type, i.e., a reference to a Quality Evidence class.

### 3.3 Dynamic model

The static model is common to all executions of the same workflow. A *dynamic model* for quality provenance, also an RDF graph, is populated during each workflow execution, and contains references to the static model. Its purpose is to capture the values of the variables involved in the workflow, i.e., those that appear in the static model, as well as the effect of the quality actions. Each new execution of the same quality workflow results in the generation of a new dynamic model (for the same static model).

From a technical standpoint, the mechanism for collecting provenance information for the dynamic model exploits Taverna’s ability to accept third party

<sup>3</sup> <http://www.w3.org/TR/owl-guide/>

<sup>4</sup> <http://www.w3.org/RDF/>

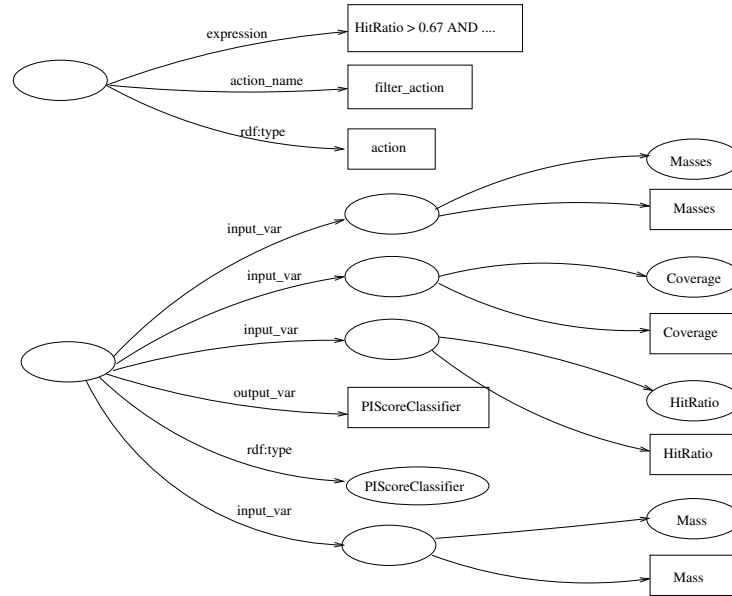


Fig. 5: Compiler-generated static model for quality provenance

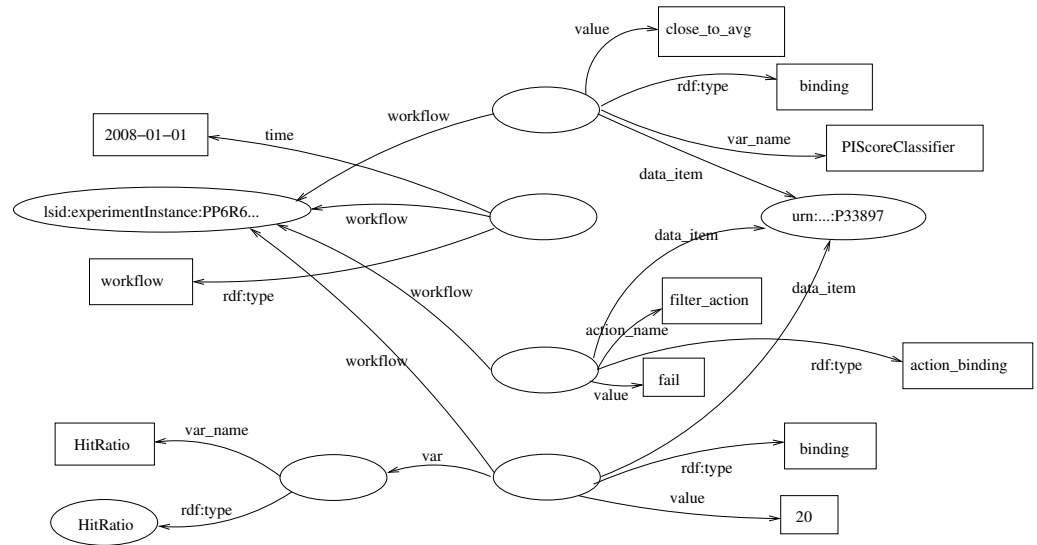


Fig. 6: Dynamic provenance model populated from workflow execution

monitoring components and to send notifications to them for a variety of events that occur during workflow execution. Using this notification pattern, the quality provenance component monitors the activity of individual processors in the quality workflow, as well as the content of the messages they exchange.

With reference to our example, Fig. 6 represents a fragment of the dynamic model corresponding to the static model of Fig. 5. The b-nodes in the middle represent a variety of workflow elements. Second from top is the workflow execution node with a unique identifier (i.e., the resource `PP6...`) that serves as a reference for the other nodes, which are related to it through the `workflow` property. This common reference defines the scope for all the resources associated with a single execution. It ensures, for example, that we can retrieve the entire quality provenance graph for one execution independently from that of other executions (using a query with the constraint that the workflow be the same for all resources returned), while at the same time allowing for queries over multiple executions, for example “all protein data in class *fail*”, simply by ignoring the workflow identifier.<sup>5</sup>

Briefly, the other b-nodes are used to represent the quality assertion value *close\_to\_avg* for a data item (top node), the quality classification *fail* for the same item (third node), and the binding of the `HitRatio` variable to value 20, still for the same item (bottom node). Note that references to the static model occur both by name, i.e., the literal `PIScoreClassifier` can be used to retrieve the quality assertion’s static information, and by reference, e.g. the `var` property for the bottom node which refers to a b-node in the static model, namely for the descriptor of the variable. A new execution of the same workflow results in a new set of b-nodes, with references to the same static model nodes.

### 3.4 Querying the model

As part of the Qurator workbench we offer a programmatic interface (in Java) for querying the model, based on the SPARQL query language (the W3C standard RDF query language<sup>6</sup>). In addition, however, we have also defined a graphical user interface, exemplified in Fig. 3 above, which implements the common types of provenance analysis listed at the end of Section 2.

The two SPARQL queries shown in Fig. 3.4 (slightly simplified for the sake of presentation) illustrate the types of provenance data retrieval supported by the model, which form the basis for the user interface. The first returns all quality classes, i.e., the outcome of action processors, for a given workflow and for each data item, while the second returns the values of all assertions, for a given data item.

---

<sup>5</sup> Recent extensions of RDF, namely for *named graphs* (<http://www.w3.org/2004/03/trix/>), can also be used to partition a large RDF graph according to a given scope.

<sup>6</sup> <http://www.w3.org/TR/rdf-sparql-query/>

```

SELECT    ?action ?outcome ?workflow
WHERE {
  ?binding data_item "P33897" .
  ?binding action_name ?action .
  ?binding value ?outcome .
  ?binding workflow ?workflow .
  ?binding rdf:type "actionBinding" .
FILTER    (regex(?workflow, "4IPQF26RXW2")) . }

```

Fig. 7: Provenance query that returns all action outcomes for a given workflow

```

SELECT    ?assertion ?value ?workflow
WHERE {
  ?binding var_name ?assertion .
  ?binding value ?value .
  ?binding workflow ?workflow .
  ?binding rdf:type "binding" .
  ?altem rdf:type "assertion" .
  ?altem var_name ?assertion .
FILTER    (regex(?workflow, "4IPQF26RXW2")) .
          (regex(?data_item, "P26153")) . }

```

Fig. 8: Provenance query that returns all assertion values for a given workflow and data item

## 4 Conclusions

In our previous work [9] we have proposed the notion of a *quality workflow* in the context of the Qurator workbench for managing information quality in e-science. In this paper we have described a focused, application-oriented process provenance model, called *quality provenance*, that can be associated to quality workflows. The model benefits from the automated generation of quality workflows by means of a compiler, and from the use of an ontology of information quality concepts – both of which are pre-existing Qurator features.

Several broad-scope provenance models have been proposed in the literature. Oblivious of any workflow semantics, these models capture a generic and low-level form of provenance. In contrast, in our approach we narrow the scope of provenance analysis, in return for the ability to present users with a high-level explanation of the processors that are within the scope. Although we have developed this idea in the context of quality-based decision processes, we believe this to be a viable approach that can be generalised to other types of pattern-based workflow structures.

## References

1. R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, 2003.
2. O. Biton, S Cohen-Boulakia, S. Davidson, and C. Hara. Querying and managing provenance through user views in scientific workflows. In *Procs. International Conference on Data Engineering (ICDE)*, April 2008.
3. S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, Cláudio T. Silva, and H. T. Vo. VisTrails: visualization meets data management. In *SIGMOD Conference*, pages 745–747, 2006.
4. A. Chapman and H. V. Jagadish. Issues in building practical provenance systems. *IEEE Data Eng. Bull.*, 30(4):38–43, 2007.
5. S. Davidson, S. Cohen-Boulakia, A. Eyal, B. Ludascher, T. McPhillips, S. Bowers, M. Kumar Anand, and J. Freire. Provenance in scientific workflow systems. In *Data Engineering Bulletin*, volume 30. December 2007.
6. C. Hedeler and P. Missier. *Database Modeling in Biology: Practices and Challenges*, ch. Quality management challenges in the post-genomic era. Artech House, 2007.

7. D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34:W729–W732, 2006.
8. P. Missier. *Modelling and Computing Information Quality in e-science*. PhD thesis, School of Computer Science, 2008.
9. P. Missier, S. M. Embury, M. Greenwood, A. D. Preece, and B. Jin. Quality views: Capturing and exploiting the user perspective on data quality. In *VLDB*, pages 977–988, Seoul, Korea, September 2006.
10. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, pages 3045 – 3054, November 2004.
11. D. A. Stead, A. Preece, and A. J.P. Brown. Universal metrics for quality assessment of protein identifications by mass spectrometry. *Molecular & Cellular Proteomics*, 5(7):1205–1211, 2006.