Quality views: capturing and exploiting the user perspective on data quality

Paolo Missier
School of Computer Science
University of Manchester
Oxford Rd, Manchester, UK
pmissier@cs.man.ac.uk

Suzanne Embury
School of Computer Science
University of Manchester
Oxford Rd, Manchester, UK
sembury@cs.man.ac.uk

Mark Greenwood
School of Computer Science
University of Manchester
Oxford Rd, Manchester, UK
markg@cs.man.ac.uk

Alun Preece
Computing Science
University of Aberdeen
Aberdeen, UK
apreece@csd.abdn.ac.uk

Binling Jin
Computing Science
University of Aberdeen
Aberdeen, UK
bjin@csd.abdn.ac.uk

ABSTRACT

There is a growing awareness among life scientists of the variability in quality of the data in both public and private repositories, and of the threat that poor data quality poses to the validity of experimental results. No standards are available, however, for characterizing and computing quality levels in this broad data domain. We argue that data processing environments commonly used by life scientists should be augmented with facilities for expressing and applying quality-based, personal data acceptability criteria. These criteria entail a trade-off between the completeness of a query result and its quality.

We propose a framework for the declarative specification of a user's personal quality processing requirements, called *quality views*. These views are compiled into a configuration of Web services that can be semi-automatically embedded within the data processing environment. The result is a quality management toolkit that promotes rapid prototyping of new quality components, and eases the reuse of existing ones. We illustrate the utility of the framework by showing how it can be deployed within Taverna, a scientific workflow management tool, and applied to actual workflows for data analysis in proteomics.

1. INTRODUCTION

Data management for the life sciences, and for post-genomic research in particular, represents a challenging testbed for data quality management. The life sciences domain is increasingly characterized by a variety of experiment types and high-throughput techniques that generate large volumes of data. In *proteomics*, for example, the study of complex biological systems requires the simultaneous identification of proteins from a sample, as well as the comparison of large numbers of samples. As large-scale automated data analysis becomes essential in order to make biological sense of the data, there is a growing awareness in the community that the data produced by the experiments is of variable quality; yet, quality of data in this domain is difficult to characterize and control.

Data quality problems concern primarily the accuracy of experimental measurements, and of their interpretation: data is produced by laboratory processes that are often experimental in nature, hence not entirely predictable nor easily reproducible. Even when they are well-established, the experiments are still subject to variability and error, due to biological contamination, procedural errors in the lab, and technology limitations [12].

Despite the general lack of quality control, the number and size of public post-genomics databases is increasing. Proteomic data can be found for instance in PRIDE[1], Pedro[11] and gpmDB[2]. This data is typically used to perform *in silico* experiments, in which hypotheses are tested computationally by analysing data from previous experiments, rather than through conventional experimental techniques in the lab. By these means, the effects of poor data quality in the primary DBs tends to propagate to the second-generation data produced by these experiments.

One of the main problems with data quality management in this domain is the lack of agreement on common quality metrics, and of practical instruments for performing quality assessments. Faced with unfamiliar data, therefore, scientists often find it difficult to answer basic questions: what are appropriate acceptability criteria for the data, and how are they computed? which types of quality indicators are

VLDB '06, September 12-15, 2006, Seoul, Korea.

[1] www.ebi.ac.uk/pride
[2] gpm – http://gpmdb.thegpm.org/

available? what levels of quality are reasonable to expect[3]? at which level of data granularity should these criteria apply?

Due to the fundamentally subjective nature of quality, the answers to these questions, when available, are inevitably based on personal heuristics and experience rather than on accepted standards. Furthermore, they are only valid for particular uses of the data. For example, scientists may guess that the reputation and track record of the originating lab for a proteomics experiment may be a good discriminator for quality. However, users seldom have a cost-effective way to validate such hypotheses, which are often expressed informally and cannot be applied to the data.

Rather than trying to identify general rules for quality assessment, we propose to provide users with tools to explore the available options and trade-offs for measuring quality. We have followed a pragmatic approach that involves real users, primarily e-scientists, who have been engaged in the definition of a common terminology and tools for information quality management. As a result of this collaboration, in the context of the Qurator project[4] we have developed a user-centred quality model and software environment, in which domain experts can rapidly and easily encode and test their own heuristic quality criteria. At the heart of the model is the novel concept of *quality view*, a sort of personalized "lens" through which the data can be viewed. Our main contributions include:

- a user-extensible semantic model for information quality concepts in e-science, which has been lacking for a long time, and which provides the basis for formulating homogeneous and shareable quality criteria;
- a process model and a simple, declarative language for the specification of abstract quality views in terms of a few elementary operators;
- an architecture for the implementation and deployment of quality views within various data processing environments.

The main difficulty in designing an environment for quality analysis in the life sciences is to make it cost-effective: it should be straightforward for scientists who are not database experts to observe their data through various configurations of quality views, but it should also be easy for them to experiment with different configurations. We have addressed this key requirement in two ways: firstly, by reducing the amount of implementation work that is required to create new quality management components and to integrate them with the user's data environment; and secondly, by increasing the possibilities for reuse of custom quality management components in future applications. This is achieved by identifying quality management functionality that is either generic across a range of analysis problems, or that is specific to an application or domain, but which can be generated automatically from a high-level specification of the user's requirements. The Qurator service-based architecture offers this common functionality and provides a uniform way to deploy user-defined, domain-specific quality services.

The choice of life sciences data as a testbed for the ideas developed in the project is key for the validation of the proposed framework; the abundance of real-life use cases provide good test suites with which to validate our approach, as our running example demonstrates.

1.1 Running example: understanding protein function

The example concerns the discovery of sets of proteins that are expressed by particular organisms or cells [1], a common problem in qualitative proteomics. One widely used technique is protein mass fingerprinting (PMF). In PMF, a sample containing a (possibly large) number of unknown proteins is processed in the lab using a mass spectrometer, in order to obtain a representation of its protein components as a list of individual masses, called a peak list. The data-intensive portion of the experiment involves using the peak list to search a reference database of known proteins, and reporting a ranked list of proteins that are likely to be present in the original sample.

Two main types of data quality problem arise in this type of experiment:

- Protein identification is intrinsically subject to uncertainty, due to limitations in the technology used, experimental contamination, an incomplete reference database, or an inaccurate matching algorithm. The results may contain false positives, and it is often the case that the correct identification is not ranked as the top match.
- Experiments performed at different times, by labs with different skill levels and experience, and using different technologies, reference protein databases and matching algorithms, are difficult to compare.

Now, consider a follow-up *in silico* experiment, that intends to exploit the results of a protein identification process, as specified by the ISPIDER project [2] on proteomic data integration. A scientist trying to understand the behaviour of a cell under particular circumstances performs a PMF experiment, from which many identifications result. Rather than the identifications *per se*, the scientist is more interested in the functional roles of the proteins within the cell. The identified proteins are therefore transformed into descriptions of their biomolecular function, by querying the GOA database, which links protein accession numbers with terms describing molecular function, expressed in a standard controlled vocabulary.[5]

This analysis involves a sequence of steps, which mixes access to software tools with database queries. It is therefore implemented using a scientific workflow editor; in this case,

[3]Generally, higher quality requirements result in less acceptable data

[4]Funded by the EPSRC Programme Fundamental Computer Science for e-Science: GR/S67593 & GR/S67609 — *Describing the Quality of Curated e-Science Information Resources.*

[5]The Gene Ontology (GO) – http://www.geneontology.org

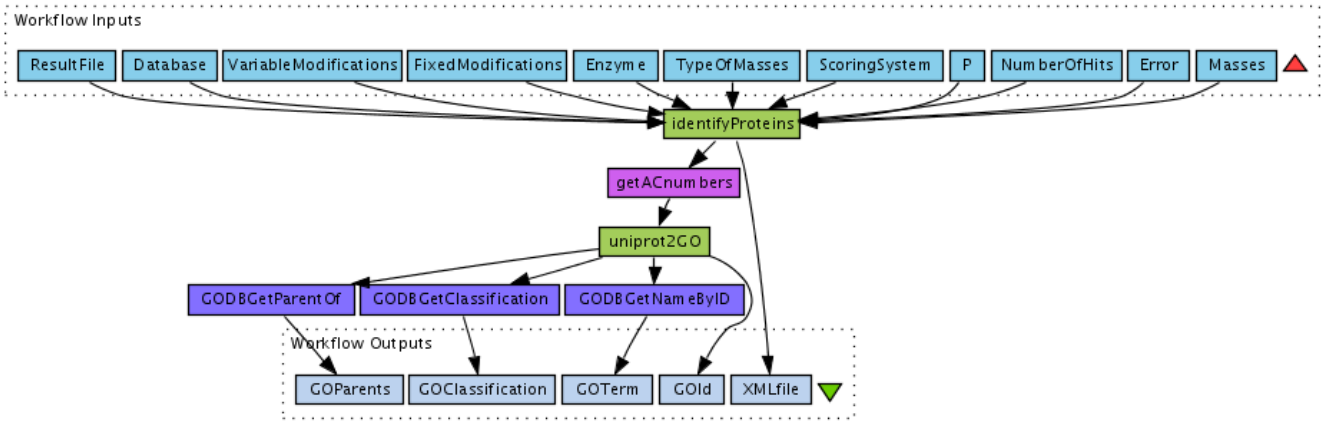

Figure 1: Example Proteomics Analysis Workflow

the Taverna workbench[6] [13], a component of the myGrid project.[7] The resulting workflow is shown in Figure 1. In the first step, a set of peak lists are retrieved from the Pedro database and used for protein identification, using the Imprint analysis tool[8] along with some configuration parameters and the reference protein sequence database. Imprint computes ranked identifications, along with additional indicators; in our example, we will use *Hit Ratio* (HR) and *Mass Coverage* (MC). HR gives an indication of the signal to noise ratio in a mass spectrum, and MC measures the amount of protein sequence matched [20]. Finally, the GOA database is queried to retrieve the functional annotations for each identified protein.

At this point, the scientist proceeds to determine the most likely protein functions, perhaps making a pareto chart of the functional annotations by frequency of occurrence, to see whether any trends emerge. What happens, however, when the protein lists include false positives? How would the scientist be able to rapidly design and repeatedly observe the effect of alternative criteria for ranking and filtering the matches? In this paper, we argue that our quality management framework provides ways to answer this question.

1.2 Related work

The research presented here is only concerned with expressing and computing quality assessment functions, rather than providing another toolkit for data cleaning, following the examples of AJAX [9], TAILOR [7], Potter's Wheel [17], and others. Comparatively little work has been done on providing user-oriented tools and languages for expressing quality in a general way; among these, the XQual language [3] is notable in its attempt to extend QML [8] for describing quality of service constraints, in order to accommodate quality of data constraints. The emphasis of the research, however, is more on the performance aspects of the quality-aware query processing than on the flexibility it affords.

In the related area of Quality of Service (QoS) specification, most of the available work on semantic modelling of QoS metrics is focused on advertising non-functional services capabilities, for quality-aware service discovery. Zhou et al. [21], for example, define a DAML ontology for QoS that is suitable for automatic matching of service quality profiles against a user's quality requirements. Thus, their notion of "service acceptability criteria", roughly corresponding to our "data acceptability criteria", reduces to service classification using ontology reasoning. Although this is a potentially interesting approach, it means that the service-matching algorithm is built-in and limited by the expressivity of the underlying ontology language. Our work differs crucially, in that we allow acceptability criteria to be defined as arbitrary decision models, rather than using ontology reasoning. Furthermore, we offer an environment in which users may define their own, customized quality processes, and observe their effect on the data. Besides simple accept/reject, these effects are described in terms of general condition/action pairs, which make more general action types possible (for instance, some data can be directed to a special workflow for dedicated processing). Finally, the greater diversity of possible quality metrics that is encountered in the data domain, suggests that pre-defined "quality profiles" are of limited use; in our approach, we instead provide a language for the dynamic composition of user-defined metrics.

An older attempt at providing a toolkit for composable quality processors is described in [5], but its development, to the best of our knowledge, has not continued. Closer to the life sciences domain, Boulakia et al.[4] have proposed a practical system for selecting biomedical data sources according to user preferences. So far, however, the search for useful quality indicators and quality functions in bioinformatics has hardly been systematic; one notable exception is an investigation into the consistency of functional annotations in the Uniprot[9] database. These annotations describe the likely function of a protein. In [16], the authors establish experimentally the reliability of a simple and readily available meta-data element, called *evidence codes*, as a possible indicator of the reliability of the curator's annotation. This type

[6]Taverna – http://taverna.sourceforge.net/
[7]myGrid – http://www.mygrid.org.uk/
[8]Imprint is an in-house software tool for PMF. A number of public and commercial tools are available, notably MASCOT [14].
[9]Uniprot – http://www.ebi.uniprot.org

of research provides us with precious experimental data regarding useful indicators that might be exploited to express new quality criteria.

1.3 Paper organization

The rest of the paper is organized as follows. After an overview of our technical approach, in the next section, we introduce models of information quality concepts in Section 3, and of abstract quality processes in Section 4. We then present our architectural framework in Section 5, and show its application to a specific user environment—in this case, the Taverna workflow environment for e-science applications (Section 6). Finally, in Section 6.3 we present an example of effective quality-aware filtering for our main case study.

2. OVERVIEW OF THE APPROACH

The definition and automated processing of quality views is based on the assumption that data can be annotated in an effective way with particular types of metadata, called *quality annotations*. Annotations may include any measurable quantity that can provide "clues" into the quality of the data, for example HR or MC, provided by the Imprint algorithm alongside the ranked list of protein IDs. We use the term *quality evidence* to refer to different types of annotation. Users may then define domain-specific functions of these annotations, called *quality assertions* (QA for short), which provide intuitive and ready-to-use expressions of quality for the underlying data in terms of data classification, or ranking. QAs are computed on a whole collection of data items, rather than on individual items. For example, given the set of protein IDs computed by one run of the Imprint algorithm, a user may define a QA that assigns a score $s(hr, mc)$ to each ID within that collection, as a function of HR and MC. Alternatively, a QA can be defined as a classifier for the collection, which associates a class label (low, mid, high) to each ID, based on the frequency distribution of the score $s(hr, mc)$. In general, different QAs, using the same or different types of evidence, capture different (and possibly contrasting) user perceptions of quality on the same data.

The Qurator quality framework lets users compose quality views that (i) compute one or more QAs, and (ii) perform actions on the data based on user-defined conditions, for example to filter out the data in class "low", or to retain the top-k data items, relative to a custom ranking computed by a QA. The framework includes the following main elements: (i) a metadata management infrastructure for computing quality annotations, maintaining a mapping from data to annotations, and retrieving annotations given their evidence type; (ii) a registry of quality annotation functions and QA functions, which are implemented as Web services, and (iii) a conditional expression language and interpreter to define and apply data acceptability actions to the data.

These low-level functionalities are not directly accessible to the users, however. Rather, they are exposed through a small collection of *quality operators*, which users may compose into abstract quality views using an XML-based language. The operators are mapped to implementation components that may execute within specific target data processing environments –in our example, a workflow language and enactment service. Thus, given a target workflow environment, the framework executes an abstract quality process by compiling it into a quality workflow, deploying, and invoking it on the data. The quality workflow can also be integrated with the user-defined workflow that generates the data.

We now introduce two models that provide a formal basis for the framework, namely (i) a semantic model of information quality management concepts, or *IQ model*, and (ii) a quality process model.

3. SEMANTIC MODEL FOR INFORMATION QUALITY

The IQ model, an ontology defined in the OWL DL[10] semantic web language, formally captures the concepts just introduced, and defines their relationships. Its root classes include Quality Assertion to represent QAs, and Quality Evidence, i.e., any measurable quantity that can be used as input to a QA, such as HR or MC. Recall that evidence is often not itself a measure of quality, but rather it enables quality assertions to be made.

The term *quality annotations* denotes the actual values for quality evidence, which are computed for specific data items by *annotation functions*, represented by classes in the Annotation Function taxonomy; these annotations are instances of ontology classes under QualityEvidence. Thus, the ontology provides both a structured vocabulary of concepts, and a schema for a knowledge base of annotations, as shown in Figure 2. Consistent with the ontology, the association between data and evidence is represented by the contains-evidence object property, with domain and range the DataEntity and QualityEvidence classes, respectively.

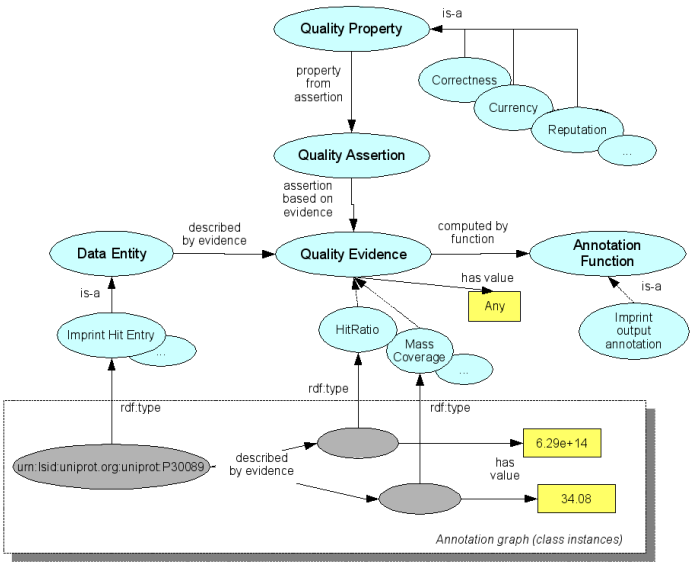

Figure 2: Fragment of IQ ontology with quality evidence annotations

The concepts rooted at the Data Entity class represent any

[10] http://www.w3.org/TR/owl-guide/

data item for which quality annotations can be computed, and quality assertions can be made. Data can be of different types and granularity, for example a single proteinID computed by Imprint (represented by the Imprint Hit Entry class), a database tuple, or an entire XML document. The association between Data Entity concepts and actual data items is maintained in a separate data model, called the *binding* model. In this model, data is represented as the result of some retrieval operation, represented generically by a *resource locator*, eg an XPath expression, or an SQL query.

The binding model is also used to map concepts under Quality Assertion and Annotation Function, to their implementation in the Web service space. These bindings make it possible to compile an abstract quality view into an executable, service-based workflow, as explained in more detail in Section 6.

Finally, the model also captures a simple collection of generic *quality properties*, or *dimensions* as they are known in the IQ literature [19, 18]; these include accuracy, completeness, currency. Users may associate quality assertions to these properties, for the purpose of classifying them and thus fostering their reuse. A more thorough description than is possible in this paper, regarding these semantic models, can be found in [15].

Annotations are encoded as a graph of RDF statements[11] (lower part of the figure), and are maintained in a dedicated repository. Note that the encoding requires that references to the data itself can be considered as resources in the RDF framework. This is achieved by "wrapping" the native data identifiers as URIs.[12] In our use cases, we have adopted the naming conventions defined by the Life Science Identifiers (LSID) initiative[13] for the unique URI-encoding of data references. Thus, in the figure, P30089 is a Uniprot accession number, the LSID-wrapper (part of Uniprot's own naming scheme) of which is the URN shown in the oval. The standard rdf:type property indicates that this is an instance of Imprint Hit Entry. The data is annotated with literal-encoded RDF values for quality evidence, that are themselves instances of model classes, namely HitRatio and MassCoverage.[14] In this case, the evidence is available as part of the Imprint output, therefore the annotation function simply captures their values and stores them as annotations. Commonly, however, annotation functions compute evidence metadata from a variety of sources; for example, when the reputation of a scientific journal is used as evidence for the credibility of published data, official impact factor tables (eg provided by the ISI[15]) may have to be consulted.

[11]RDF – http://www.w3.org/RDF/

[12]It is assumed that unique identifiers are available throughout for data items (in bioinformatics databases, these are commonly known as *accession numbers*).

[13]Life Sciences Identifiers Specification, Object Management Group (OMG), document dtc/04-05-01

[14]In the model, we exploit the flexibility of the RDF model to allow for values of quality evidence that are themselves arbitrary RDF graphs; however, this feature is not discussed further in the paper.

[15]http://www.isinet.com/

4. ABSTRACT QUALITY PROCESSES

Users exploit the IQ model to create a *quality process*, which are executed to reach in acceptability decisions on their data. Two main elements are available for decision making. The first is a set of one or more QAs which capture alternative data quality preferences by computing classifications and scores for the data. The second is a set of condition/action pairs, where conditions are predicates on the values of QAs and of the evidence. These may involve filtering data items based on their class, or more generally, partitioning the data so that the different subsets can be handled independently. Take for example our protein ID classification function. Having partitioned protein IDs into classes (low, mid, high), users may now experiment with different filtering conditions, eg "select the *high and mid* IDs for which the Mass Coverage is also greater than X".

The distinction between the QA and actions decision steps is mainly a pragmatic one. Using QAs, arbitrary heavy-weight decision models can be encoded, for instance complex decision trees, but no actions take place other than "tagging" the data with a class label, or a score. QAs are expected to be well-tested, and reusable – such is the case for the QA functions used in our example, which are backed by scientific experimental evidence [20]. Action conditions, on the other hand, can be modified on-the-fly, from one process execution to the next, allowing users to quickly observe the effect of various filtering options.

From the user perspective, quality assessment involves the composition of a quality process, and its repeated execution, possibly using different action conditions. Specifically, quality process execution amounts to (i) collecting the quality evidence associated with the data, required to compute the QAs, e.g. HR; (ii) computing the QA functions using the input quality evidence, and (iii) evaluating the quality conditions and executing the associated actions.

The process is sketched in Figure 3, where a data set is partitioned into several subsets, according to the outcome of the classification process. As shown in the figure, the process of collecting quality evidence can be broken down into two steps, namely (i) computing new metadata values using annotation functions; and (ii) retrieving previously computed values that have been stored in a metadata repository. This distinction is motivated by the observation that, although annotations may in principle be generated on the fly, in some cases this is neither necessary nor convenient. When the quality process involves querying a database with stable data, for example, then the quality annotations are likely to be long-lived and can be made persistent. Take for instance the Uniprot database; a measure of measures of credibility of a functional annotation made by a Uniprot curator, whether based on the evidence codes to which we alluded earlier or other evidence, is bound to be long-lived, relative to the execution of a query to Uniprot. On the other hand, caching annotations is not an option when the evidence is produced as part of the same process that computes the data, as in our Imprint case study, where the scope of annotations is a single process execution. Therefore, in practice the actual annotation process may consist of both on-the-fly computation of annotations, and simple retrieval from a repository.

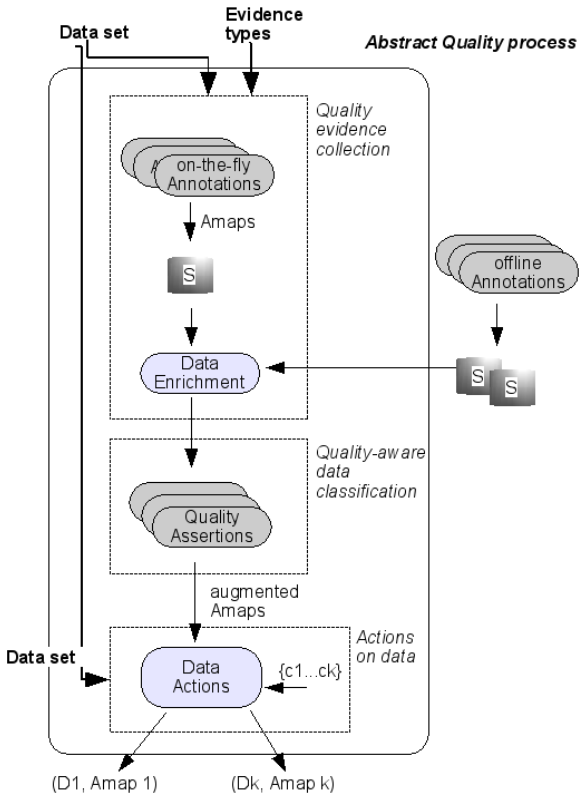

Figure 3: General quality process pattern

Figure 3 depicts a generic quality process pattern, composed using several abstract operators. We will now describe the role of the operators, and then show how they are mapped to an extensible set of Web services in the Qurator framework; this provides an instantiation of the pattern as a process that can be deployed within the user environment.

4.1 Quality operator types

We adopt the following terminology for describing the quality operator types shown in Figure 4. Given a data set D and a set E of evidence types, an *annotation map*: $Amap : d \rightarrow \{(e, v)\}$ associates an *evidence value* v (possibly null) for evidence type $e \in E$ to each data item $d \in D$. The names or evidence types must be consistent with the IQ ontology; specifically, they must be references to subclasses of QualityEvidence, eg HitRatio.

We also use mappings of the form $\{d \rightarrow (t, cl)\}$ to represent the assignment of class cl to d within a classification scheme t. This mapping is produced by a QA operator. Again, t is a reference to a subclass of ClassificationModel, say PIMatchClassification, and cl is a member of that model, eg average-to-low.

Quality Assertion. An operator of this type defines a decision model that associates a class value to each data item based on the contents of a vector of evidence values, represented by an input annotation map. These operators are expected to be user-defined and domain-specific. However, to the extent that the decision model only depends on the evidence and not on the data itself, they are not specific to an individual data set: they can be applied to any data set that can be annotated with the input evidence types.

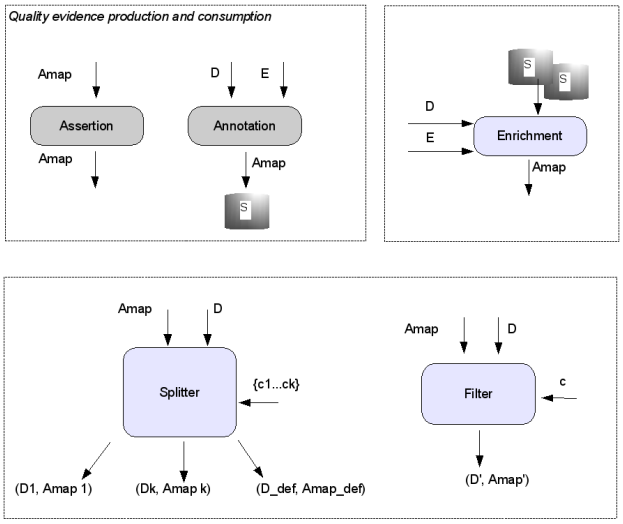

Figure 4: Abstract quality process operators

The operator computes a new version of its input map, augmented with new mappings for the class assignment, of the form $\{d_i \rightarrow (t, cl_j)\}$.

QA operators are defined in the IQ ontology as the subclasses of the QualityAssertion class. Thus,

$$\text{UniversalPIScore} \subseteq \text{QualityAssertion}$$

is a domain-specific operator of this type. Note that operators are defined as classes rather than individuals; this is done to allow further user-defined specializations on the operators' hierarchy.

Annotation. This operator computes a new association map of evidence values for an input set E of evidence types, and for each item in the input data set D. The map is stored persistently in a repository s specified as part of the input.

In practice, this operator may require additional input to execute, e.g. the species of a protein (human, mouse...); we assume that the operator has access to this additional input, which is not explicitly represented in the quality process.

Similar to QAs, these operators are also user-defined; however, note that not only they are domain-specific, but they are also data-specific, hence they offer few opportunities for reuse besides their repeated application to homogeneous data sets.

Data Enrichment. This operator type accounts for the need to fetch pre-computed annotations from a repository, given an input data set D and a set E of evidence types of interest. It is pre-defined and not user-extensible. Since a metadata repository may contain annotations for a large number of data items and for a variety of evidence types, this operator effectively performs queries to the repository, using $d \in D$ and $e \in E$ as lookup keys.

Actions. These operators evaluate boolean expressions on evidence and quality classification values, and assign data items to different groups accordingly. The expression language includes relational operators, i.e. "score < 3.2", as well as set membership operators, as in "PIScoreClassification IN {"high", "mid"}". More examples are provided in Section 5.1, when the declarative specification of quality processes is presented. The set of possible actions is extensible; we consider only two example here:

Data splitting action. This action type splits an input data set D into groups $D_1 \dots D_k$, not necessarily disjoint. The input consists of D, an annotation map $Amap$ (which may include classification mappings), and a collection $\{c_1 \dots c_k\}$ of conditional expressions over the quality evidence types defined in $Amap$. The output consists of $k+1$ sets of pairs $(D_i, Amap'_i)$, such that for each $d \in D$, map entries $\{d \rightarrow (e_1, v_1), \dots, d \rightarrow (e_n, v_n)\} \subseteq Amap$ are placed in $Amap'_i$, and d is added to D_i, if and only if $c_i(v_1 \dots v_n)$ evaluates to true. The $k+1$-th output is a default group, which includes all data items and associated evidence, for which none of the expressions evaluates to true.

Data filtering action. This is a particular case of data splitting, for which a single condition c is given and a single output map is produced; the map entries that satisfy c are placed in the output map, while the others are simply discarded.

The Qurator quality framework, described next, provides an execution environment for quality processes in which these abstract operators are implemented as services.

5. A SERVICE-BASED QUALITY FRAMEWORK

A high level view of the framework appears in Figure 5. The core components are shown in the "Qurator services" box. These are web services that implement the user-extensible set of QA and Annotation operators, and are recorded in a service registry. In order to facilitate the process of binding the abstract operators to the services, all QA services export the same WSDL interface, using a common XML schema for the input and output messages. The schema is effectively a concrete model for the data sets, evidence types and annotation maps described earlier in abstract terms. Among the core services, we find the Data Enrichment operator, already described, as well as further ancillary functionality that will not be discussed in this paper.

On the right hand side, the data layer includes a collection of quality annotation repositories that contain quality evidence metadata, according to the model discussed in Section 3. All of these repositories are accessed through the same read/write API, which provides an object model for building and searching the annotation graphs, and guarantees that the metadata complies with the ontology model. Consistent with the definition of annotation maps, access to evidence is provided primarily based on (data, evidence type) keys, using queries in the SPARQL language, currently a W3C working draft.[16] While performance issues have not been addressed at this stage, it is worth noting that the use of SPARQL makes it simple to swap the underlying storage mechanism and/or back-end database, should performance become a concern. Scalable RDF storage components are currently offered for instance by the Sesame project[17] and by Oracle [6].

5.1 Quality views: declarative specification of quality processes

On the left side of Figure 5 we find the quality view management services that exploit the framework by building on the core services. Quality views are concrete and machine-processable specifications for instances of our general quality process pattern, expressed in an XML syntax. They include declaration sections for quality operators and evidence types, which can be referenced within the scope of the specification using explicit variable names. Although expressed using a concrete syntax, views are still defined purely in terms of our abstract model, i.e., the specification is not tied to any implementation of the operator set. This leaves us free to target the view to different data management environments. In the next section, we demonstrate this targeting step by showing how a view process instance can be deployed within the Taverna workflow environment.

As an illustration of a quality view we have extended the workflow shown in Figure 1 which was developed by the ISPIDER proteomics project. The fragment below declares an annotation operator, of class q:Imprint-output-annotation, and gives it a local variable name, q:ImprintOutputAnnotator[18]. The <variables> declarations identify the evidence types for which the operator is going to provide values, which are to be stored in the cache repository; the persistent attribute is set to 'false', indicating that these annotations are only valid during one process execution.

```
<Annotator
    serviceName="ImprintOutputAnnotator"
    serviceType="Imprint-output-annotation">
    <variables repositoryRef="cache"
      persistent="false">
      <var evidence="q:Coverage"/>
      <var evidence="q:Masses"/>
      (...)
    </variables>
</Annotator>
```

Three QAs are used in this view, so that users may compare their relative effects by editing the selection criteria in the action section at process execution time. The first two QAs produce a score based on a combination of Hit Ratio and Mass Coverage, and Hit Ratio alone. The scores can be used to split the data, using a splitter action based on user–defined thresholds. To simplify the user's task, a third QA computes a ready-to-use three-way classification (low, mid, high) based on the average and standard deviation of the

[16] SPARQL – http://www.w3.org/TR/rdf-sparql-query/. Several RDF store providers offer SPARQL support, e.g. 3store (http://www.aktors.org/technologies/3store/), see also [10].

[17] SESAME – http://www.openrdf.org/

[18] 'q' is a prefix for the namespace http://www.qurator.org#

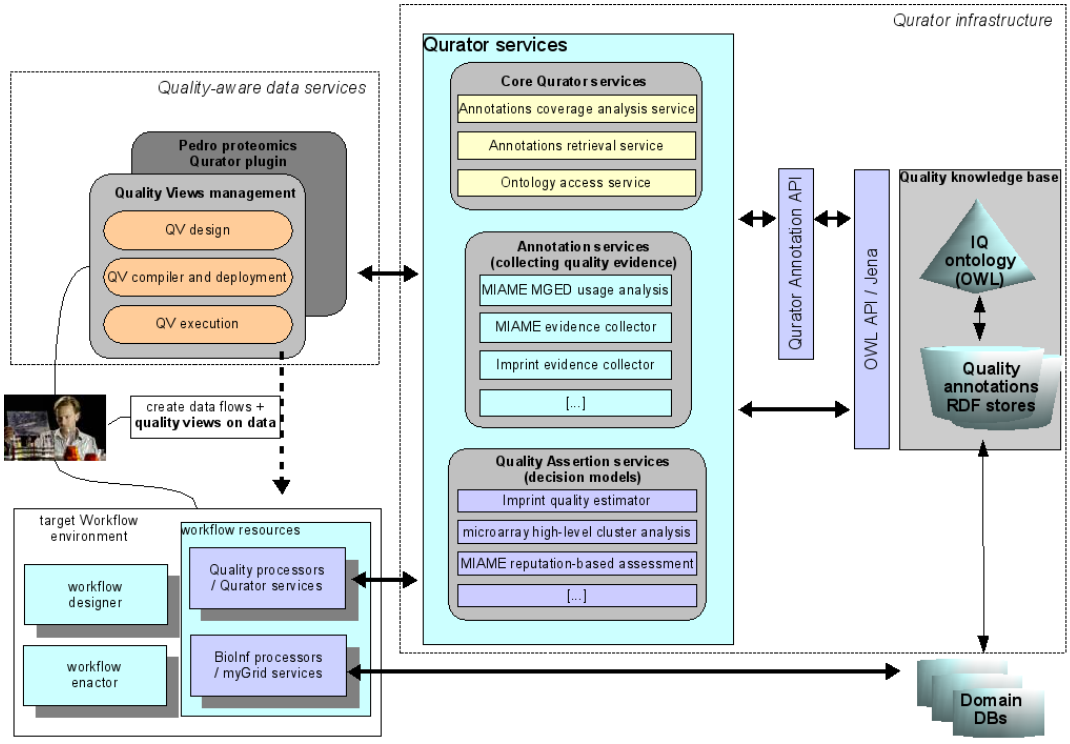

Figure 5: Overview of the quality framework

Hit Ratio and Mass Coverage score.[19]

The following fragment shows the declaration for the first QA:

```
<QualityAssertion
    serviceName="HR_MC_score"
    serviceType="q:UniversalPIScore2"
   tagName="HR_MC"
    tagSynType="q:Score">
    <variables repositoryRef="cache">
      <var variableName="Coverage"
           evidence="q:Coverage"/>
      <var variableName="Masses"
           evidence="Masses"/>
      <var variableName="PeptidesCount"
           evidence="q:PeptidesCount"/>
    </variables>
```

It defines the HR_MC_score operator as an instance of class q:UniversalPIScore2, which computes a score value and associates it to tag name HR_MC, a variable. Its three inputs are instances of the mentioned QualityEvidence subclasses, and are fetched from the cache repository, which has been written to by the previous annotation operator. A classification QA is declared similarly:

```
<QualityAssertion
    serviceName="PIScoreClassifier"
    serviceType="q:PIScoreClassifier"
    tagSemType="q:PIScoreClassification"
    tagName="ScoreClass"
    tagSynType="q:Class">
    <variables repositoryRef="cache">
      (...)
    </variables>
</QualityAssertion>
```

Here, q:PIScoreClassification is the IQ model class that defines the classification schema used by the QA. We should clarify that views may include any number of annotator and quality assertions operators, which may fetch their annotations from any number of repositories.

The variable names introduced in the preceding sections can now be referenced in one ore more action sections, where splitters and filters are specified:

```
<action name="filter_top_k_score">
<filter>
<condition>ScoreClass in
''q:high'', ''q:mid''
and HR_MC > 20
</condition>
</filter>
</action>
```

Note that the classifications computed by the QA are themselves defined as part of the IQ ontology: they are enumerated individuals of class q:PIScoreClassification.

[19]The thresholds used for classification are (avg - stddev) and (avg + stddev).

View specifications do not include any reference to input data sets, because they are designed to be independent of the specific input data. The run-time model for quality views descends directly from our definition of the quality operators (Section 4.1): a view is applicable to any data set for which evidence values are available for the required evidence types mentioned in the input.

6. COMPILING AND EMBEDDING QUALITY VIEWS WITHIN SCIENTIFIC WORKFLOWS

The process of targeting a quality view to a specific environment requires two additional types of information: (i) a set of *bindings* of abstract operator types to implemented services, and (ii) deployment instructions for the target environment.

The binding information is maintained in a semantic registry whose schema is defined in a binding model, mentioned earlier. This small ontology prescribes a pattern for associating any concept defined in the IQ ontology with a concrete Service Resource or Data Resource object through a Binding object. A Resource has a locator associated with it, whose nature depends on the type of the resource, eg a service endpoint. We use this ontology to bridge the gap between the conceptual model and the framework implementation: the binding step results in each Annotation and (QA) operator being mapped to a Web Service endpoint.

6.1 Quality view compilation into a workflow

To make our case study concrete, our example of quality view compilation is based on the Taverna workflow environment, rather than on some abstract workflow reference model. The simple workflow design primitives offered by Taverna, however, are common to many similar models, and the approach can easily be generalized. In Taverna, processors drawn from an extensible collection[20] can be composed using either data or control links. A control link from processor A to B means that B is started as soon as A completes. The workflow execution environment invokes the processors and transfers data from the processors' output ports to input ports according to a simple data model.

Adding new processors to the available collection is straightforward, as any deployed Web Service with a published WSDL interface can be found automatically on a specified host by Taverna's *services scavenger* process. As a preliminary step, core Qurator services, as well as user-defined annotation and QA services, have been added to Taverna's processor collection. The main rules for compiling quality views specifications into a workflow are as follows:

- Annotators are added first; their input ports are initially unbound, and their output is empty, since annotators only write to a repository;
- By analysing the annotators and QA specifications, the QV compiler determines the association between each evidence type and the repository in which its value is to be found. This allows the compiler to add one single Data Enrichment (DE) operator and configure it using this association, to make it read from specific repositories. A control link is also installed from each of the annotators to the DE;
- The output from the DE, an annotation map, feeds all the QA processors, using their common WSDL interface;
- Action processors are added next, and data connectors are installed from each of the QAs to each of the actions. During deployment (see below), the output ports of actions are bound to data links that transfer the surviving data back to the embedding workflow.

[20]More than a hundred services are currently available for Taverna, most of them for bioinformatics applications.

The compiled workflow for our running example appears in Figure 6, box (a); the ConsolidateAssertions task is added by the compiler to produce a consistent view of multiple assertions; a number of other ancillary tasks, used to encode configuration information for the main Taverna processors, are not shown.

6.2 Deployment descriptors for embedding

The homogeneity of the quality and data process models make the embedding of one workflow within another a conceptually simple operation. Two main elements must be considered, (i) a set of adapters that surround the embedded quality flows, and (ii) the connections among host and embedded processors, which may occur through the adapters.

The Taverna-specific deployment descriptor contains declarations (using a succinct XML syntax) for both adapters and connectors. Adapters typically account for differences in data formats; as they are Taverna processors themselves, their names are registered and can be used within the descriptor. Connectors include the name of the source and target processors and the name of the output and input ports, respectively. Figure 6 shows the original experiment workflow from the running example, with the embedded quality workflow. Note that, at this point, both the annotator and the DE processors are provided with an input (the data set and the quality evidence of interest); also, the protein identification task feeds the quality view through the adapter, and the output of the filter feeds the GO retrieval task.

6.3 Applying quality views in practice: some example results

We now present some results that were obtained by applying our embedded quality view to the protein identification flow from the running example. While this shows the practical usefulness of our approach, the results presented here should be viewed only as an illustration of the potential offered by our framework, as no effort was made in this experiment to reach biologically significant conclusions.

The overall effect of inserting a quality process into the original flow is to reduce the number of protein IDs, so that the associated GO terms more accurately reflect the likely functions of the protein. As we recall from Section 1.1, the ISPIDER workflow takes a set of masses from a mass spectrometer and delivers a set of terms from the GO ontology, based on the GO annotations of the identified proteins.

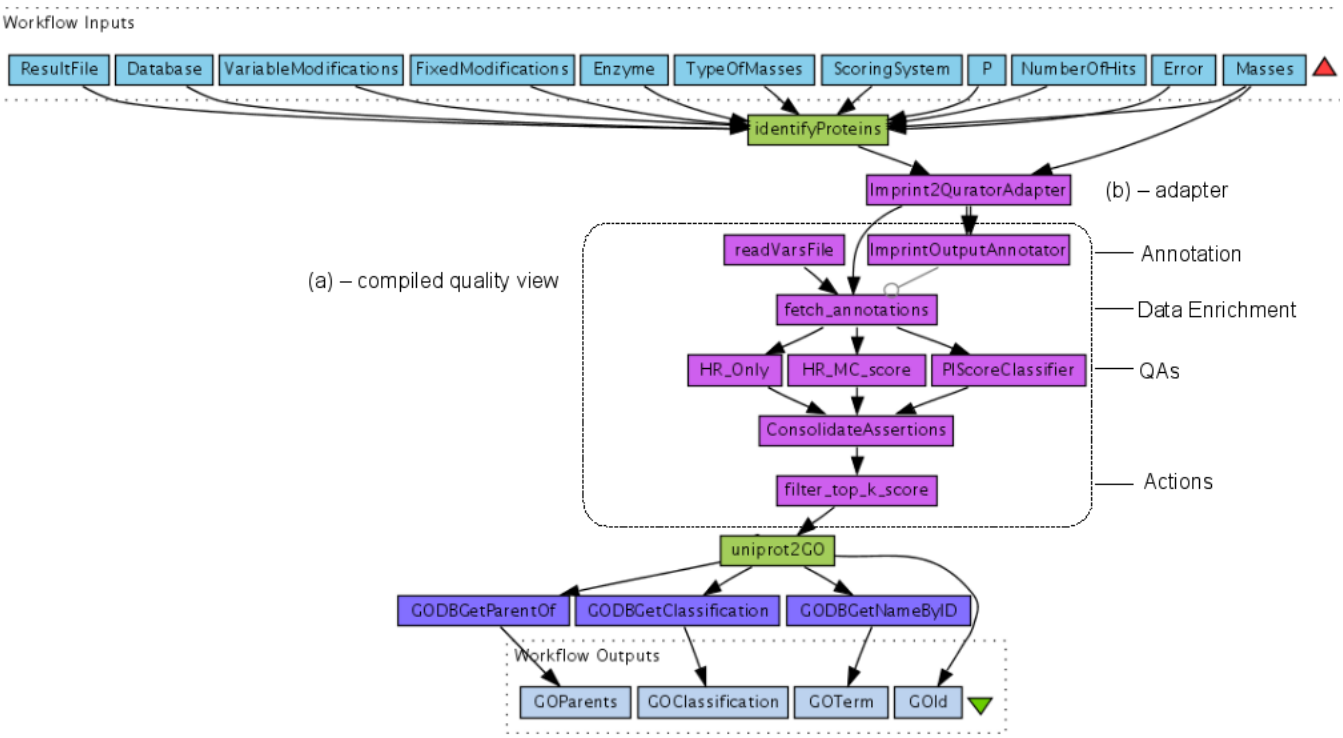

Figure 6: Compiled and embedded quality workflow

While some GO terms may occur very frequently when they are accumulated over the entire experimental sample (many spots), their frequency may not be an accurate indicator of their relevance because the GO terms associated to the false positives are misleading and should be discounted.

In this experiment, protein IDs are classified according to our QA criteria, regardless of the native ranking proposed by Imprint, thus providing an ad hoc quality view of the associated GO terms.

Input for the ISPIDER workflow includes the peptide masses for 10 protein spots, extracted from a PEDRo data file supplied by collaborators in the Molecular and Cell Biology Group, School of Medical Sciences, University of Aberdeen. The 10 sets of peptide masses were processed using the original ISPIDER workflow, producing a total number of about 500 related GO terms. They were then processed again by adding the quality workflow shown in Figure 6, with a filter action set to save only the "top quality" protein IDs, i.e., those with a score higher that the average + standard deviation (see Section 5.1).

As a measure of actual significance for the GO terms, we take the ratio of the number of occurrences of each GO term with and without quality filtering. Figure 7 shows the resulting GO terms, ranked in order of significance as just defined (the original rank is not shown). A high ratio indicates that the GO term is relatively unaffected by the filtering, and thus it is representative of high-quality proteins. This significantly alters the original ranking: for instance, GO term GO:0007049, now ranked first, occurred only 6 times in the original data, while GO:0008652, ranked towards the end, originally occurred 14 times.

7. CONCLUSIONS AND FURTHER WORK

We have presented the Qurator framework for the composition of user-defined quality functions into abstract *data quality views*. Views can be compiled, embedded within specific data processing environments, and computed during the execution of the data processes. We have given an example of this mechanism in action, using a proteomics process based on Taverna, a scientific workflow environment which is in wide use among the bioinformatics community.

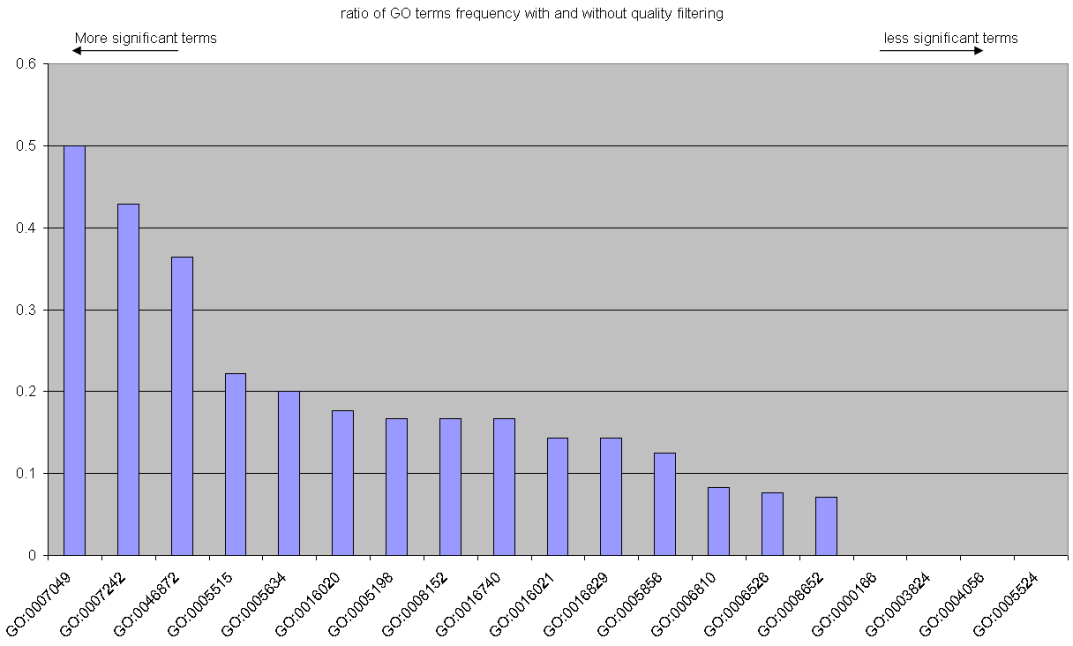

Figure 7: Effects of a data quality view on the workflow output

With this exercise, we have begun to validate our hypothesis that the framework supports cost-effective data quality management, by identifying various levels of sharing and reuse, namely: (i) of quality concepts through the IQ model, (ii) of generic core framework components, (iii) of configured components for a whole data domain. The limits of reuse are also becoming clear: some components, primarily used for extracting quality evidence, tend to be very data-specific.

Life sciences applications continue to provide new requirements for the architecture and real use cases for testing our implementation. Our current work mainly involves (i) engaging biologists in the definition of a useful collection of quality functions, (ii) investigating the use of machine learning techniques to derive decision models and quality functions from example data sets, (iii) providing a more general mapping from quality views to formal workflow models, and (iv) providing user-friendly interfaces for the reuse of quality components views defined by peers within a scientific community.

Acknowledgements

The authors would like to thank Dr. David Stead of the Molecular and Cell Biology group, School of Medical Sciences, University of Aberdeen for precious insight into quality functions for proteomics data, and Dr. Khalid Belhajjame for providing support with ISPIDER experiments.

8. REFERENCES

[1] R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422:198–207, March 2003.

[2] K. Belhajjame, S.M. Embury, H. Fan, C. Goble, and al. Proteome data integration: Characteristics and challenges. In *Proceedings of UK e-ScienceAll Hands Meeting*, 2005.

[3] Laure Berti-Equille. Quality-adaptive query processing over distributed sources. In *Procs. 9th International Conference on Information Quality, ICIQ 2004, Cambridge, Ma*, 2004.

[4] S. Cohen Boulakia, S. Lair, N. Stransky, S. Graziani, F. Radvanyi, E. Barillot, and C. Froidevaux. Selecting biomedical data sources according to user preferences. In *ISMB/ECCB 2004, Bioinformatics*, volume 20, suppl. 1, pages I86–I93, 2004.

[5] F. Caruso, M. Cochinwala, U. Ganapathy, G. Lalk, and P. Missier. Demonstration of telcordia's database reconciliation and data quality analysis tool. In *VLDB 2000, September 10-14, 2000, Cairo, Egypt*, pages 615–618. Morgan Kaufmann, 2000.

[6] E. I. Chong, S. Das, G. Eadon, and J. Srinivasan. An Efficient SQL-based RDF Querying Scheme. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P. Larson, and B. C. Ooi, editors, *VLDB 2005, Trondheim, Norway, August 30 - September 2*, pages 1216–1227. ACM, 2005.

[7] M.G. Elfeky, A.K. Elmagarmid, and V.S. Verykios. Tailor: a record linkage tool box. In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, San Jose, CA, Feb. 2002. IEEE Computer Society.

[8] S. Frolung and J. Koistinen. QML: A language for Quality of Service specification. Technical Report HPL98-10, HP Labs, HP Software Technologies Laboratory, 1998.

[9] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An Extensible Framework for Data Cleaning. In *Proceedings of the 16th International Conference on*

Data Engineering (ICDE 2000), San Diego, CA, USA, 2000.

[10] S. Harris and N. Gibbins. 3store: Efficient bulk rdf storage. In *Proceedings 1st International Workshop on Practical and Scalable Semantic Web Systems*, Sanibel Island, Florida, USA, 2003.

[11] K. Garwood K, T. McLaughlin, C. Garwood, and al. PEDRo: a database for storing, searching and disseminating experimental proteomics data. *BMC Genomics*, 5(1), Sep 2004.

[12] H. Muller, F.Naumann, and J.C. Freytag. Data quality in genome databases. In *Proceedings of the Eight International Conference on Information Quality (ICIQ03)*, Cambridge, MA, 2003. MIT.

[13] 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.

[14] D.N. Perkins, D.J.C. Pappin, D.M. Creasy, and J.S. Cottrell. Probability-based protein identification by searching sequence databases using mass spectrometry data. *Electrophoresis*, 20:3551–3567, 1999.

[15] Alun D. Preece, Binling Jin, Edoardo Pignotti, Paolo Missier, Suzanne M. Embury, David Stead, and Al Brown. Managing information quality in e-science using semantic web technology. In York Sure and John Domingue, editors, *ESWC*, volume 4011 of *Lecture Notes in Computer Science*, pages 472–486. Springer, 2006.

[16] P.W.Lord, R.D. Stevens, A. Brass, and C.A.Goble. Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–83, 2003.

[17] Vijayshankar Raman and Joseph M. Hellerstein. Potter's wheel: An interactive data cleaning system. In *VLDB*, pages 381–390, 2001.

[18] T.C. Redman. *Data Quality for the Information Age*. Artech House, 1996.

[19] Wang R.Y. and Strong D.M. Beyond accuracy: What data quality means to data consumers. *Journal of Management Information System*, 12(4), 1996.

[20] D. A. Stead, A. Preece, and A. J.P. Brown. Universal metrics for quality assessment of protein identifications by mass spectrometry. *Molecular & Cellular Proteomics*, 2006. In press. Available at http://www.mcponline.org/papbyrecent.shtml.

[21] Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. DAML-QoS Ontology for Web Services. In *ICWS*, pages 472–479. IEEE Computer Society, 2004.