

Semantically Annotated Provenance in the Life Science Grid

Bin Cao, Beth Plale, and Girish Subramanian

School of Informatics and Computing
Indiana University, Bloomington, IN, USA
{plale, bincao, subramag}@cs.indiana.edu

Paolo Missier and Carole Goble

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

Yogesh Simmhan

Microsoft Research
One Microsoft Way
Redmond, WA, USA
yoges@microsoft.com

Abstract— Selected semantic annotation on raw provenance data can help bridge the gap between low level provenance events (e.g., service invocations, data creation, message passing) and the high-level view that the user has of his/her investigation (e.g., data retrieval and analysis). In this initial investigation we added semantically annotated provenance to the Life Science Grid, a cyber-infrastructure framework supporting interactive data exploration and automated data analysis tools, through (i) automated data provenance collection and (ii) automated semantic enrichment of the collected provenance metadata. We use a paradigmatic life sciences use case of interactive data exploration to show that semantically annotated provenance can help users recognize the occurrence of specific patterns of investigation from an otherwise low-level sequence of elementary interaction events.

Keywords- life sciences, provenance, semantic annotation

I. INTRODUCTION

Cyber-infrastructure frameworks for experimental science are becoming an increasingly popular way of interacting with a variety of analysis tools and other computational and data resources on the Internet. Automated provenance [6] metadata, collected during the course of a scientist's interaction with the framework during a data exploration session, can add value to the exploration process in a number of ways: it can be used to reproduce analyses and processes, identify the causality of a series of events, broaden sharing and reuse of data products, support the long-term preservation of scientific data, attribute ownership, and determine the quality of a particular data set. Raw provenance data, however, consists mainly of observations of a user's interaction with some visual interface, as well as of system-level observations of system events (service invocations, data creation, message passing). Unlocking the potential of such provenance metadata requires bridging the gap between these low level events, and the view that the user has of his/her investigation, which is likely to be described in terms of high-level information processing, typically consisting of data retrieval and analysis steps that lead to some scientific finding. The work described in this paper stems from the hypothesis that augmenting raw provenance metadata with selected semantic annotations helps bridge this gap, and furthermore, that for the most part such annotations can be obtained automatically, i.e., with minimal user effort.

We explore this hypothesis in the specific context of the Eli Lilly open source Life Science Grid (LSG) [3], a cyber-infrastructure framework built from Microsoft .NET 2.0 Component Application Block (CAB) and Web Services that couples automated data visualization and display (through the CAB) with invocation of data sources and analysis tools (through Web Services). The LSG is in production use inside Eli Lilly with a more fully functioning open source version anticipated.

We approach the study by defining a paradigmatic use case for interactive exploration of life sciences data, and used it to drive the design of an architectural model that integrates LSG with (i) automated data provenance collection, using the Karma provenance framework [7] developed at Indiana University, and (ii) automated semantic enrichment of the collected provenance metadata, using the Semantic-Open Grid Service Architecture (S-OGSA) semantic annotation framework [1] developed at University of Manchester. The use case is based on the data playground idea, first proposed by Gibson et al. [2], which builds on the hypothesis that recognizable patterns of a complex data exploration process may emerge from the continuous observation of direct user interaction with data exploration and analysis tools.

The remainder of the paper describes an initial investigation into the potential for the use of provenance in this scenario, specifically to help users recognize the occurrence of specific patterns of investigation from an otherwise low-level sequence of elementary interaction events. Thus, in addition to describing the use case (Section II) and presenting the technical architecture that made this investigation possible (in Section III), we reflect upon the type of provenance metadata that can be usefully and inexpensively collected, semantically annotated, and exploited to add value to scientific findings.

II. USE CASE

The use case driving our work, shown in Figure 1, describes a realistic scenario of exploratory analysis on genes and gene products. The example is representative of a typical investigation method in bioinformatics, where a small set of genes that are known to be involved in a particular disease, in this case human diabetes, is used as a seed to grow a larger collection of related genes, which will provide the scope for further and possibly more expensive lab analyses. The collection grows incrementally, in a series of iterations where

a gene pool, indicated as the *working set* in Figure 1, is updated by either adding or removing some of its elements. The iteration involves a combination of access and user interaction with public databases accessible through web services on the web (we use the NCBI Entrez service for searching gene details and the AmiGO browser for Gene Ontology associations), and the use of Basic Local Alignment Search Tool (BLAST) in order to reveal homologous genes in model organisms, typically the mouse. Genes obtained from BLAST are again inspected by the user, and can be selected for addition to a “working set” maintained on behalf of a user, or discarded based on the user’s judgment. The process can repeat possibly multiple times, by again BLAST-ing some of the mouse genes, leading to a larger pool of human genes that are more or less directly related to each other through homology properties.

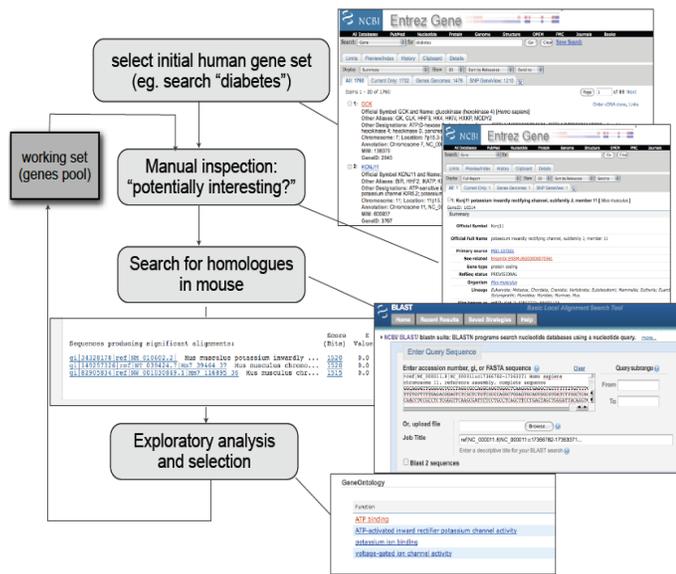


Figure 1. Illustration of use case

Throughout this process, users interact with a variety of interfaces, which LSG integrates into one single visual environment, as described in the next section. Although the iterations indicate a logical sequence of events, users are not constrained by any prescribed course of action; indeed, most of the steps can be performed in any sequence, making for a variety of different analysis paths. At the end of the process, it is important for users to understand how a certain final working set of genes was accumulated: certain genes were discovered but discarded, others were deemed worthy of further investigation, others were first added and then replaced by other, more promising elements. A combination of raw provenance metadata, user-provided and automatically added semantic annotations is used to support the explanation process. Raw provenance includes a trace of all the invocations to services through the LSG interface, as well as all UI interactions. User-provided annotations include optional descriptions that explain each update decision that affected the working set (addition, removal), and semantic annotations are obtained from various sources, for example a registry of semantically annotated Web Services, as described in the next Section.

III. SYSTEM ARCHITECTURE

We view the use case as an instance of a general user interaction model, where events and data products are recorded and associated to a *user session*, and various annotations are associated to both the events, for example a service invocation, and the data products, e.g., the result message from the service (we define a session as being delimited by user login and logout actions). Figure 2 gives an overview of the architecture used to support this interaction model. Individual users can configure their own personal LSG desktop environment by selectively enabling some of the available plugins, which control the interaction with specific services. In addition, LSG plugins interact amongst each other using a publish/subscribe model through an LSG event bus, providing users with an integrated, multi-panel interface. Thus, suppose for example that an NCBI Entrez¹ plugin accepts user gene lookup requests, and sends the corresponding gene descriptions onto the bus, while an AmiGO² plugin that is able to resolve Gene Ontology (GO) terms subscribes to those descriptions. When the user submits a request, the response triggers the AmiGO plugin, which responds by updating its own interface with the GO descriptions of the gene, while details of the latter are being displayed on the NCBI Entrez plugin interface.

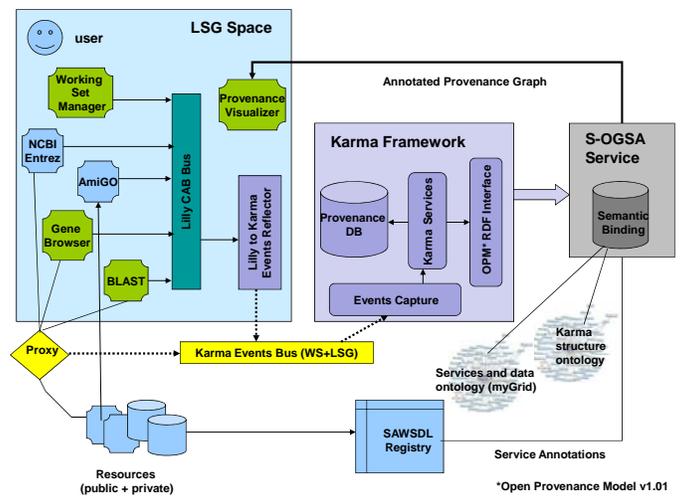


Figure 2. Integrated provenance management architecture

We have exploited this event model to generate elementary provenance events through the Karma component, which is configured to snoop on the LSG event bus, in addition to having its own instrumentation in the Web Service proxies that mediate LSG plugin interactions with the services. Karma structures these provenance events according to the Open Provenance Model (OPM) [5], a

¹ NCBI Entrez is a search engine for biomedical databases and available at

<http://www.ncbi.nlm.nih.gov/sites/entrez?db=gen>.

² AmiGO is a web service to access Gene Ontology associations for genes and available at

<http://www.geneontology.org/>.

community standard for describing causal graphs through a set of pre-defined types of nodes and their relationship.

Throughout a user session, fragments of OPM graphs representing single interactions are forwarded to the S-OGSA component, which performs two functions: firstly, it analyses the OPM graph and adds semantic annotations to some of its nodes, whenever possible and by using a variety of annotations sources. For instance, if a node represents a Web service invocation, and a semantically annotated description of the service is available, then S-OGSA augments the OPM graph by associating the annotations to that node (a more detailed description of the annotation architecture is described in Section C). Secondly, S-OGSA stores the pair <user session, OPM graph> in its own database, which the Provenance Visualizer can query to present semantic provenance to the user. By having the Provenance Visualizer implemented as a new LSG plugin itself, the combination of these components provides users with a seamlessly integrated feedback loop, by incrementally displaying the effect of their actions as a rendering of provenance metadata.

Next, we elaborate on the three main components of the integrated architecture.

A. LSG

Two main features make LSG an appealing platform for our experimentation: its openness, which made it possible to create provenance events simply by adding a subscriber to the LSG event bus, as described earlier; and its extensibility, which we have used to implement new plug-ins especially for our use case. Specifically, we have used two of the available plugins for the open source version of LSG, namely for searching the NCBI Entrez database and for resolving GO terms; and have implemented three new plugins:

- A BLAST plugin that interacts with one of the several publicly available BLAST services³;
- A Working Set Manager, to manage the dynamic collection of data products, in this case genes, that represent the main outcome of the users' investigation;
- A Provenance Visualizer, which can display parts of the provenance graph to the users (see Figure 3).

The "LSG Space" in Figure 1 shows the relationship amongst these plugins. While we exploit the event model to automate much of the data flow across the plugins, we also identify points in the process where we felt that explicit, knowledge-intensive user input was desirable. Thus, for example, while it is possible to extract a DNA sequence in FASTA format from an NCBI gene description record, to be used as input to BLAST, expert users prefer to have control over the portion of the sequence, for example to include or exclude the gene promoter regions on either side of the sequence. This mix of automated data flow and explicit user input offers the additional opportunity for users to add their

own notes as explanations of their actions, for example to comment on the choice of a wider region around a gene. This is particularly clear in the design of the Working Set Manager, which automatically accepts new elements, i.e., genes from BLAST, through the event bus, but also offer users the opportunity to examine (accept, reject, annotate) each of them individually.

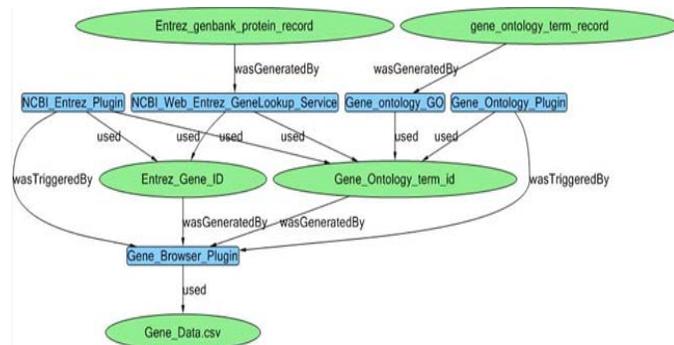


Figure 3. OPM graph fragment

B. Karma and OPM

The main functions of the Karma component in this setting are to capture raw provenance events, and to format them according to the Open Provenance Model specification. As Karma is a general provenance collection and management tool, it implements a generic provenance model and set of instrumentation tools that are independent of the application system. Instrumentation of the LSG required the use of several forms of instrumentation. For the web hosted data services and sources, we implemented proxy web services that utilize instrumentation handlers in Axis2 to collect provenance. Provenance of the CAB activity is captured by a listener on the CAB events bus. The listener forwards provenance relevant events to Karma. The high level view of capture is shown in Figure 4.

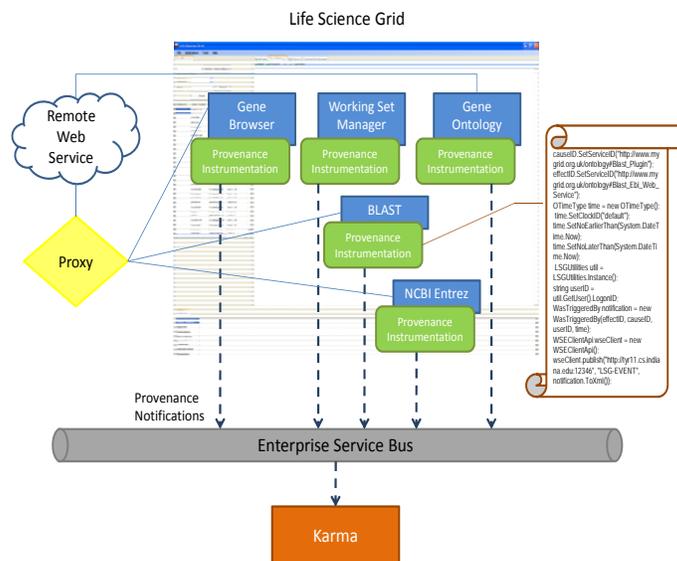


Figure 4. Provenance instrumentation in LSG plugins

³ <http://www.ebi.ac.uk/Tools/blast/>.

We distinguish between black box plugins, for which it may be possible to observe data exchange events that occur through the LSG bus, and white box plugins, where in addition, user interaction events that occur through a service interface can also be detected. In practice, black boxes are those where native web pages are displayed, so that access to the user click-throughs on the page is limited and can only be achieved by intercepting the HTTP requests using a proxy, for instance, but some of the context in which the request is made is missing. In white box components, on the other hand, the UI is part of the plugin design, and as a consequence we can capture user actions with full detail.

According to the black-box, white-box distinction, the Working Set manager is a white box, because all user events can be observed along with optional user annotations, while the native LSG plugins, the NCBI Entrez and AmiGO plugins, are black boxes. As for the BLAST plugin, making it a white box required the extra effort of encoding a bespoke web-based interface to interact with the service, in order to capture all of the important user interactions. Thus, Karma captures the user selection, de-selection, and annotations of genes in Working Set Manager, and the set of genes that transit on the LSG bus, including BLAST reports.

Karma maps provenance events to fragments of OPM graphs. In its simplest form, an OPM graph consists of two types of nodes, which represent Artifacts and Processes. These are shown as ovals and rectangles, respectively, in Figure 3. Nodes are connected using directed labeled arcs, which express properties that hold between two nodes. The set of all legal properties is fully described in [5], however the following three types of properties were found to be sufficient to express our provenance events:

- process P *used* artifact A, for example, NCBI_Entrez_Plugin used Entrez_Gene_ID,
- artifact A *wasGeneratedBy* process P, for example, Entrez_Gene_ID wasGeneratedBy Gene_Browser_Plugin, and
- process P1 *wasTriggeredBy* process P2, for example, NCBI_Entrez_Plugin wasTriggeredBy Gene_Browser_Plugin.

The first two properties express ordinary producer/consumer relationships, while the latter is useful in expressing the indirect interaction between two plugins that publish and subscribe to a data element, respectively. We also use the same property to express the fact that a plugin controls an underlying service, i.e., in the typical situation where a service invocation is triggered by a plugin.

Provenance events are published as notifications to the Web services-based message broker, *WS-Messenger* [8], where Karma is a subscriber. When a provenance notification arrives, the corresponding provenance handler picks it up, retrieves the raw provenance data, and stores these data into its own provenance database, a MySQL relational database. These raw provenance data can be used to answer general provenance questions as well as determine the artifact dependency and the process dependency during a

user session. Meanwhile, these data is sent to S-OGSA for semantic annotation. Since OPM is an abstract process model with multiple concrete serialization formats for portability across applications, as indicated in Figure 2, we have used the RDF⁴ serialization to transfer OPM graph fragments from Karma to the S-OGSA component.

C. Modular Semantic Annotations using S-OGSA

S-OGSA [1, 4] manages the persistent and stateful associations between Grid resources, i.e., data or services, and their annotations (or any form of related metadata), expressed primarily as RDF graphs. Such associations, known as semantic bindings, can be queried with SPARQL. S-OGSA mapping to this project has user sessions playing the role of resources, with OPM provenance graphs produced by Karma as their associated metadata. S-OGSA additionally augments the input graphs with semantic annotations. Here we focus on the latter part of the S-OGSA architecture⁵.

The annotation architecture is based on the principle that annotations to nodes in the RDF OPM graph will depend on (i) the specific types of Artifact and Process nodes, and (ii) the availability of metadata sources that can be used to derive interesting metadata for those node types. To account for this flexibility, we designed a modular architecture based on the interceptor pattern, consisting of an extensible chain of annotators, each specialized to annotate specific types of nodes. Each annotator receives an input RDF graph, produces an augmented version of the same graph with annotations added to it, and forwards it to the next annotator down the chain. As no parts of the input graph are ever removed, annotators can be added incrementally to S-OGSA, in a monotonic fashion. The pattern is illustrated in Figure 5. As a proof of concept, we have implemented a chain consisting of two annotators, one for Process node of type Web Services, and one for Artifact nodes of type Blast report. We now describe how each of these two annotators uses a different metadata source to produce its annotations.

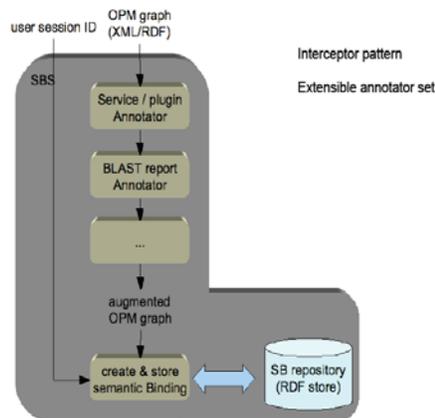


Figure 5. S-OGSA interceptors for incremental semantic annotations of OPM graphs

⁴ <http://www.w3.org/RDF/>

⁵ Technically, S-OGSA relies on the Anzo RDF API for storing its annotation graphs, and its functionality is exposed as a RESTful Web Service.

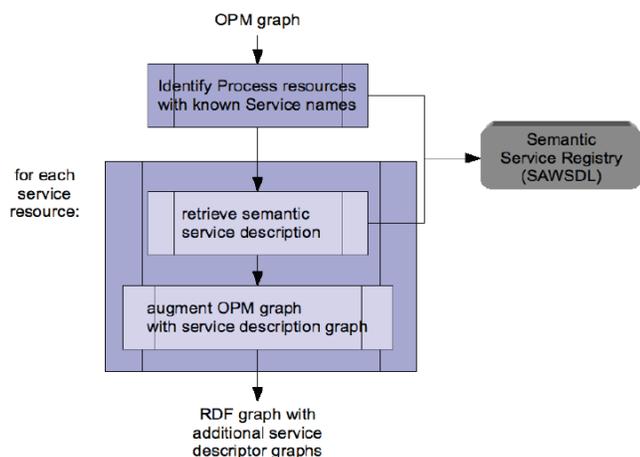


Figure 6. Service annotation

The *Service Annotator* relies on Process nodes that represent Web Services, to be labeled with a service name, for instance NCBI_Entrez, that can be matched against a local and bespoke registry of Web service descriptions. In this registry, service descriptions are semantically annotated using SAWSDL⁶ (in a future version, the Biocatalogue service registry⁷ will be used for this purpose). If a match is found, the corresponding SAWSDL annotations (i.e., the sawsdl:modelReference attribute values), are added to the RDF graph (see Figure 6). Since these annotations are references to concepts in some ontology (expressed as URIs), the standard rdf:type property is used to associate the annotation to the Process node. An example of SAWSDL-annotated service description for NCBI Entrez is shown below.

```
<wsdl:interface name="eFetchGeneService"
sawsdl:modelReference="http://www.mygrid.org.uk/ontology#E
ntrez_GenBank_protein">
  <wsdl:operation name="run_eFetch"
    pattern="http://www.w3.org/ns/wsdl/in-out"
    sawsdl:modelReference="http://www.owl-
ontologies.com/unnamed.owl#run_eFetch_dbGene">
    <wsdl:input element="nsef:eFetchRequest" />
    <wsdl:output element="nsef:eFetchResult" />
  </wsdl:operation>
</wsdl:interface>
```

Note that this entry annotates a generic NCBI eFetch service with concepts from the myGrid ontology⁸, which qualify it as a gene lookup service.

The *Blast Report Annotator* is an example of data annotator that performs complex lookups in multiple public databases in order to semantically annotate a data entry of a specific type in the OPM graph. Its general structure is shown in Figure 7. The fragment above the line is part of LSG processing. The BLAST report is accessible to the

⁶ <http://www.w3.org/2002/ws/sawsdl/>

⁷ <http://www.biocatalogue.org/>

⁸ <http://www.mygrid.org.uk/tools/service-management/mygrid-ontology/>

annotator through a unique ID that is part of a RDF resource (a URI), and that is dereferenced against a persistent local data store.

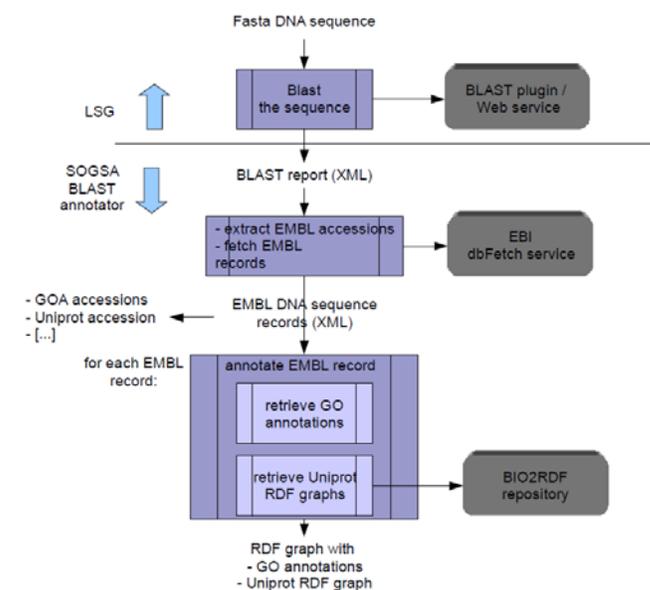


Figure 7. BLAST report annotation

An EBI BLAST report consists of a ranked list of matched DNA sequences, which may be parts of genes or proteins. Thus, some of these entries may optionally contain a variety of references to external databases; in our implementation we have focused on (i) Uniprot accession numbers, which appear whenever the matched DNA sequence is related to a protein, and (ii) GO annotations, i.e., references to entries in the Gene Ontology.

As the report is in a standard XML format, the annotator begins by extracting the EMBL accession numbers, which are then used to query the EMBL database, through the WSDbFetch Web Service⁹. This yields one XML document for each hit in the BLAST report, indicated as “EMBL DNA sequence records” in the figure. Then, for each of these records the annotator extracts both the set of GOA annotations (in the example: {A6NMx8, Q09428}), and the set of Uniprot accession numbers, if any (in the example: {Q09428}). The former is used to query the Gene Ontology to retrieve the associated descriptions, while we use the latter to query Bio2Rdf (using the dynamic URL <http://bio2rdf.org/uniprot:Q09428>). This is particularly interesting, as the Bio2Rdf project (<http://bio2rdf.org/>) exposes the content of entire Bioinformatics databases, including Uniprot, as RDF graphs. Thus, associating the RDF entry for a specific protein, when available, is a very natural operation in the context of Blast report annotation.

Figure 8 shows a fragment of annotated RDF graph for a BLAST report. The content of each report resource is a bag of entries, i.e., a bag of b-node resources, each corresponding to one sequence hit in the report. All b-nodes have type EMBLRecord (a class in the myGrid ontology) and have an

⁹ <http://www.ebi.ac.uk/Tools/webservices/services/dbfetch>

associated (a) EMBL accession number, (b) GOA accessions, if any, and (c) entire named graphs that resolve to the Uniprot records associated to the sequence, if available. Note that this type of graph could not be obtained automatically from an RDF-based provenance capture engine such as that of the Taverna workflow system¹⁰.

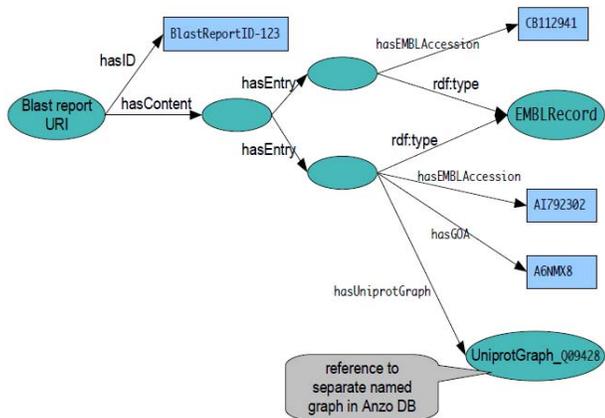


Figure 8. Annotated RDF graph for a BLAST report

IV. LESSONS LEARNT AND FUTURE WORK

This paper details the architectural complexity of collecting provenance data from LSG, augmenting it, semantically annotating it, and returning it to the user. The paradigmatic use case shows that semantically annotated provenance can help users recognize the occurrence of specific patterns of investigation from an otherwise low-level sequence of elementary provenance events.

There are several noteworthy outcomes that emerged in implementing the use case we describe in this paper. The Life Science Grid, as mentioned earlier, is built using the .NET Component Application Block “portal” with call-outs to web services. The CAB is multicast in that plugins drop events on a bus that are picked up and acted upon by all other plugins creating a stateless, shared-all medium. But once the architecture is extended to include web services that gather information from public databases and services on the web, even for our straightforward use case, the need for state sharing arises. State can be viewed as an artifact (in OPM terms), and one that is important to the provenance record particularly where long term preservation of data is the goal. We will more fully examine this impact in future work.

Substantial investigation remains in the visual presentation of provenance information. Lilly sees the historical, lineage nature of provenance as having significant potential to contribute to the drug discovery process. We are exploring visualization of higher levels of abstraction of provenance to more closely match user’s investigative process. It raises interesting questions on the implication to instrumentation as well. When can the underlying low level

event collection be replaced with higher levels of abstraction and what form do these higher levels of instrumentation take? Moreover, a user study can assess the value that provenance collection brings to the daily research investigative process of the users. We are working with Eli Lilly to set this up.

Finally, provenance can be captured and semantically annotated for other grid systems such as caGrid [9], the service-based infrastructure that supports the cancer Biomedical Informatics (caBIG¹¹). Unlike user-driven (streaming) workflows in LSG, caGrid users need to pre-define a workflow using a workflow orchestration tool before execution [10]. Since Karma can capture provenance from different service-based sources, through proper instrumentation in the workflow orchestration tool, raw provenance data can be captured and then semantically annotated by S-OGSA.

ACKNOWLEDGEMENTS

Our thanks to Katy Wolstencroft and Dr. Andy Gibson at University of Manchester for their help in the formulation of the usage scenario. Susie Stephens and Richard Bishop of Eli Lilly have been instrumental drivers in the project.

REFERENCES

- [1] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, C. Goble, “An Overview of S-OGSA: A Reference Semantic Grid Architecture,” *Journal Web Semantic*, vol. 4, no. 2, pp. 102–115, 2006.
- [2] A. Gibson, M. Gamble, K. Wolstencroft, T. Oinn, C. Goble, K. Belajjame, P. Missier, “The Data Playground: An Intuitive Workflow Specification Environment,” *Future Generation Computer Systems*, vol. 25 no.4, pp. 453-459, April 2009.
- [3] Life Sciences Grid, <http://sourceforge.net/projects/lsg/>
- [4] P. Missier and P. Alper and O. Corcho and I. Dunlop and C. Goble, “Requirements and Services for Metadata Management,” *Journal IEEE internet Computing*, Special issue on Semantic-Based Knowledge Management, Sept. / Oct., 2007.
- [5] L. Moreau, B. Plale, S. Miles, C. Goble, P. Missier, R. Barga, Y. Simmhan, J. Futrelle, R. E. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludaescher, N. Kwasnikowska, Jan Van den Bussche, T. Ellkvist, J. Freire, P. Groth, “The Open Provenance Model (v1.01),” July 17, 2008. <http://eprints.ecs.soton.ac.uk/16148/1/opm-v1.01.pdf>
- [6] Y. Simmhan, B. Plale, and D. Gannon, “A Survey of Data Provenance in e-Science,” *ACM SIGMOD Record*, vol. 34, no. 3, pp. 31-36, 2005.
- [7] Y. Simmhan, B. Plale, and D. Gannon, “Karma2: Provenance Management for Data-Driven Workflows,” *International Journal of Web Services Research*, vol. 5, no. 2, pp. 1-22, 2008.
- [8] Y. Huang, A. Slominski, C. Herath, D. Gannon, “WS-Messenger: A Web Services-Based Messaging System for Service-Oriented Grid Computing,” *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 166 – 173, 2006.
- [9] J. Saltz, S. Oster, S. Hastings, S. Langella, W. Sanchez, M. Kher, P. Covitz, T. Kurc, K. Shanbhag, “caGrid: design and implementation of the core architecture of the cancer biomedical informatics grid”, *Bioinformatics*, vol. 22, no. 15, pp. 1910-1916, June 2006.
- [10] W. Tan, P. Missier, R. Madduri, I. Foster, “Building Scientific Workflow with Taverna and BPEL: A Comparative Study in caGrid,” *Service-Oriented Computing --- ICSOC 2008 Workshops*, pp. 118-129, 2009.

¹⁰ The Taverna provenance component is now being re-defined using a non-RDF data model.

¹¹ <https://cabig.nci.nih.gov/>