A multi-system provenance architecture based on PROV-DM for Explainable AI: Application Track

Nicholas J. Car
SURROUND Pty Ltd &
Australian National University
nicholas.car@surroundaustralia.com

Robert A. Atkinson
SURROUND Pty Ltd &
Open Geospatial Consortium
rob.atkinson@surroundaustralia.com

Abstract

SURROUND Australia Pty Ltd is a small technology company focused on explainable AI and knowledge management products. At the core of our company operations is standardised provenance tracking using the PROV Data Model (PROV-DM). Using standardised provenance throughout our company lets us both assure customers that any results we produce for them are explainable, regardless of the specific systems we employ, and also allows us to supply provenance tooling to clients that will work well with other logging or provenance systems.

We generate PROV-DM-based provenance in all our important business workflows by using either our in-house *ProvWF* tool or workflows within our proprietary *SURROUND Ontology Platform* tool. We implement PROV-DM within our Knowledge Graph (KG) data products, which are a major part of our business, through the *SURROUND Ontology Platform* also and we "hand off" provenance tracking of some objects to the Git version control system. Our various workflows' and KGs' provenance information is used together to provide explainable AI results.

1 Introduction

SURROUND Australia Pty Ltd ("SURROUND") is a small technology company founded 6 years ago. While aiming to supply mainstream AI and knowledge management products to government and private sector markets, SURROUND attempts to distinguish itself from competitors through sophisticated use of Semantic Web data due to the belief that such data is the form that best preserves meaning over time and system & organisational change. Specific value propositions for SURROUND's customers, based on Semantic Web data use, are:

- the expressive power of RDF Schema and OWL2 [1, 6] for complex data modelling
- the ability to reuse many existing, sophisticated, publishd ontologies directly
- the extensibility of RDF graph-based data structures
- the systems-independence of Semantic Web data formats
- the ability to use a Semantic Web layer to act as a bridge between internal, siloed applications
- the ability to share data across organisational boundaries with no inter-organisational special data contracts (due to semantic modelling of all data elements)
- the data validation power of modern constraints languages such as SHACL [2]
- the advanced reasoning capabilities of OWL & SHACL

Emergent from some of these points is SURROUND's ability to provide comensurate provenance information across all our different systems due to them all implementing the PROV Data Model (PROV-DM) [4] in its ontology form, PROV-O [3], and our ability to produce PROV-O data, to share it between applications, to store it and present it.

In this paper we make no new research claim - this is an *Applications Track* paper - but we do aim to show "innovative use of provenance" and "the deployment of provenance-based solutions" that indicate a certain maturity of approach to the use of provenance for operational tasks. We expect this will be of interests to our industry peers and to academics wishing to know where industry implementers are up to in order to consider next research steps.

We will overview our specific company-wide provenance systems, discuss two projects that use them, indicate why we've chosen certain PROV-related implementations over others and where we think some of the provenance standards need enhancement for our purposes.

2 Simple enough theory, complex practice

Extraction of useful information present within heterogeneous or large-scale data contexts may be performed in several ways. If some of the data has some known structure then queries

can be used to select relevant subsets. The trivial form of this is searching against text content, with various degrees of sophistication. These may involve statistical techniques to identify patterns in the data. SURROUND uses Machine Learning (ML) approaches to train systems to correlate or discover information based on various patterns. We also use Semantic or Knowledge Graph-based contextual information to assist such processes to improve performance. Conversely, we also use ML approaches to infer structure. Some of our projects include Human-in-the-loop (HITL) activities too, to refine, record and improve training of systems. Performing these tasks requires us to implement complex, hybrid systems that use reasoning and ML to create ways to to organise and retrieve information from complex projects, two of which are summarised in the next sections.

The role of provenance for us within these systems is twofold: firstly so that we can provide our customers certainty of project outputs, and scondly so that we can track our systems performance so we can improve our offerigs over time. The challenge is that many, very different, types of systems interact, and so to provide end-to-end provenance for customers and ourselves that is usable, we have to pay close attention to the provenance models use across all of them and their technical integration.

From the theoretical perspective, this is simple and facilitated by PROV-DM acting as canonical model for provenance which SURROUND applies "everywhere" (as far as we practically can). The practical challenges are still significant though and can be characterised as being of two main types:

1. **Shared entity identification** - making sure different systems can use the same identifiers for entities as they pass between systems

2. **Granularity** - having useful levels of detail about pieces of knowledge whilst retaining ability to have overviews from the process level and support large scale batch processing

Shared entity identification is well handled using the RDF meta model, which uses unique URIs for things (entities and others) and the Open World Assumption, as multiple systems can represente knowledge in RDF data structures and then join knowledge by referencing shared URIs. SURROUND not only communicates our provenance infromation in RDF but, for most projects, our primary data also. Since our primary data, perhaps electronic records described using RDF metadata, and our provenance information describing that data's generation are both represented in RDF, we can identify entities across both information holdings too, not just across sytems within each holding.

Architecturally, our primary project data an our provenance can be integrated across multiple subsystems if:

1. Object identity is established as part of KG management and accessed via APIs from this shared body of knowledge when required

2. Object identity is managed within the KG whenever "human in the loop" interactions are required

3. Processing subsystems preserve and report canonical object identities

4. Coherent sets of objects may be managed in specialised persistence systems, as long as the description of data set itself and access mechanisms are part of the knowledge graph (data set granularity)

5. All processing reports provenance along with outputs using a canonical model

6. Processing elements are identified in the knowledge graph in a coherent way

Figure 1 shows the user interface of the *SURROUND Ontology Platform* (SOP) displaying provenance information within a sankey diagram. The provenance information was generated according to PROV-DM/PROV-O by SURROUND's processing workflow tool *ProvWF* which, in the displayed instance, performed Named Entity Recognition against electronic records using cloud-hosted scalable services and some of SURROUND's Knowledge Graph products for entity matching. In addition to displaying the provenance information in particular ways, SOP also managed is in bundles as *Managed Graphs* whihc, from SOP's point of view are yet another semantic asset for which provenance (and ownershi, access etc.) is automatically stored.

The provenance granularity issue identified above can be summarised by examining a range of different types of processing that may typically occur in a heterogeneous system, perhaps one performing ML to augment a knowledge base. Table X

3 Company-wide provenance architecture

SURROUND has a company-wide policy on provenance which is simple that *all* project data processing and system operations for clients - work on demand or services supplied - must be recorded in PROV-O-compliant forms, with the exception of code and data stored in version control systems which, in all instances so far, are implemented in Git[1]. So far, this policy has been implemented for all our main projects' systems and processes, but not for minor projects, some legacy systems or administrative support functions.

[1]Git is a free and open source distributed version control system, see https://git-scm.com/

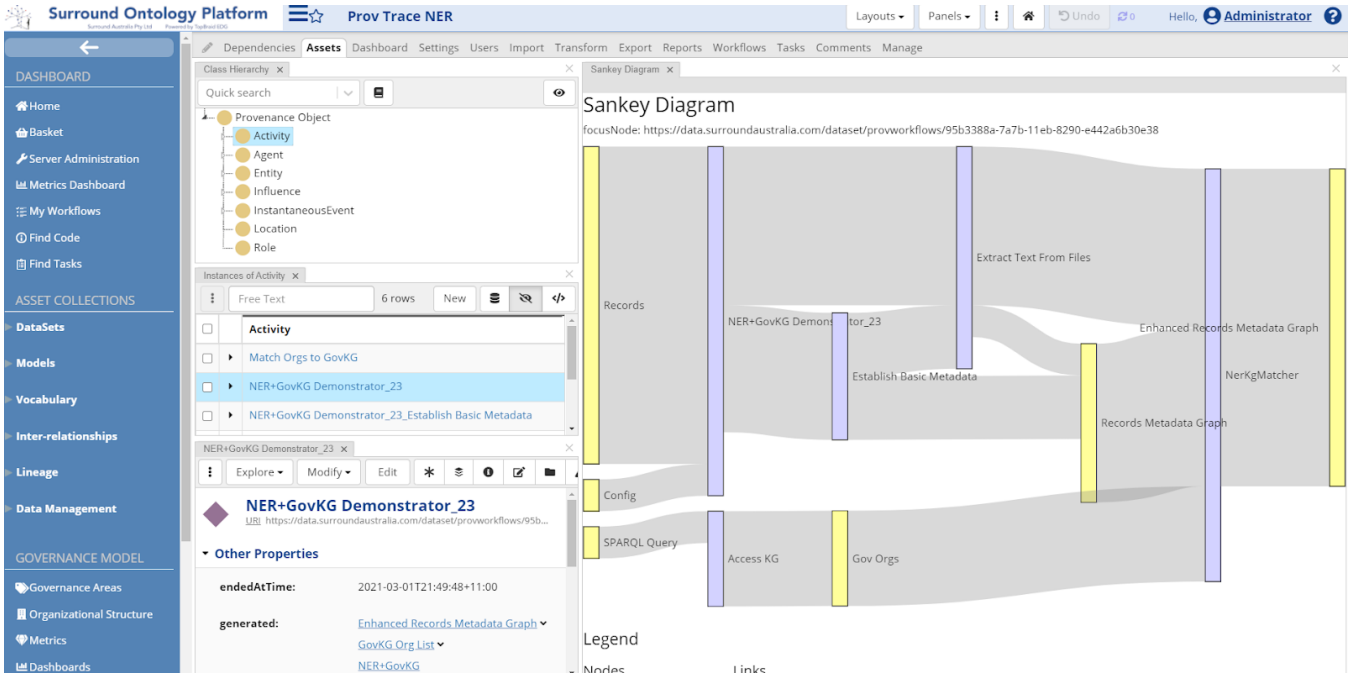

Figure 1: An example of a provenance trace from a processing workflow that uses elements of a knowledge graph, performs processing in cloud-hosted scalable services, generates augmented views of an input stream (performing Named Entity Recognition on a document set and annotating with elements from the knowledge graph), persists the results in the knowledge graph and integrates the provenance trace with the provenance trace generated by knowledge graph management.

Figure 2 links types of IT project assets we work with for clients to the tools we implement to record PROV-O-compliant provenance.

Our major provenance tools, as named in Figure 2, and the actions they perform are:

- **SURROUND Ontology Platform** (SOP)[2]
  - an enterprise data management system based on sematic data that is based on Top Quadrant's *EDG*[3]
  - SOP extends EDG adding the ability to manage various types of semantic assets and collections of them
  - SOP records PROV-DM provenance for all semantic asset actions
- **ProvWorkflow** (ProvWF)[4]
  - a Python framework and library for the creation of workflows
  - records PROV-DM provenance for all actions performed by the workflow and all data consumed or produced
  - supported by SURROUND's *Block Library*
- **Block Library**
  - SURROUND's catalogue of ProvWF *Blocks* which are PROV-DM `Activity` class objects
  - the library stores many reusable functions within *Blocks*, such as Knowledge Graph API requests, NLP text processing etc.
- **Git**
  - we use Git to track the versions of many assets - code, data etc. - in both public and private repositories
  - although we know of Git-to-PROV mapping tools, e.g. Git2PROV [5], we have not yet found it necissary to materialise such mappings but instead record identifiers for versions of PROV-DM `Entities` using Git and refer to them in PROV-O data
-

In addition to these provenance-specific tools, we implement provenance tracking with general-purpose tools too, such as:

[2] https://surroundaustralia.com/sop
[3] https://www.topquadrant.com/products/topbraid-enterprise-data-governance/
[4] https://surroundaustralia.com/provwf

Function	Examples	Granularity
Human-in-the-loop	Establishment of definitions, Registration of individual entities, Annotation, Classification for training	Statement, Reified statements
Database management, Data transformation	Making a set of data instances available to a knowledge base in a useful form	Data set (table, spreadsheet, graph etc)
Query Governance	Extraction of data subsets	Data set, Result set Data set
Bulk document processing	Indexing, classification, clustering	Container (directory, database, storage bucket etc)
Document analysis	Making information elements in a document available to finer grained processes	Document, derived data set
Knowledge Graph Management	Establishment of a known state of complex, modular knowledge graphs, Capture of provenance, Support for automated updates	Graph (Data set)

Table 1: Blabla

- **RDFlib** (SOP)[5]
  - a general-purpose RDF manipulation library, written in Python
  - many of our data objects are RDF graphs
  - used to create reified provenance for RDF statements

We ensure provenance information generated by one of our tool is usable in another, for example, *ProfWF*'s outputs can, and often are, stored in *SOP* which is then able to visualise that provenance either in isolation order by joining it to other provenance information, perhaps captured by *SOP* itself.

Availability

USENIX program committees give extra points to submissions that are backed by artifacts that are publicly available. If you made your code or data available, it's worth mentioning this fact in a dedicated section.

References

[1] Dan Brickley and R.V. Guha. RDF Schema 1.1. W3C Recommendation, World Wide Web Consortium, February 2014.

[2] Holger Knublauch and Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, W3C RDF Data Shapes Working Group, 2017.

[3] Timothy Lebo, Satya Sahoo, and Deborah McGuinness. PROV-O: The PROV Ontology. W3C Recommendation, W3C Provenance Working Group, 2013.

[4] Luc Moreau and Paolo Missier. PROV-DM: The PROV Data Model. W3C Recommendation, World Wide Web Consortium, 2013.

[5] Tom De Nies, Sara Magliacane, Ruben Verborgh, Sam Coppens, Paul Groth, Erik Mannens, and Rik Van de Walle. Git2PROV: Exposing Version Control System Content as W3C PROV. In *Proceedings of the 12th International Semantic Web Conference*, volume II. Springer.

[6] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation, 2012.

[5] https://github.com/RDFLib/rdflib

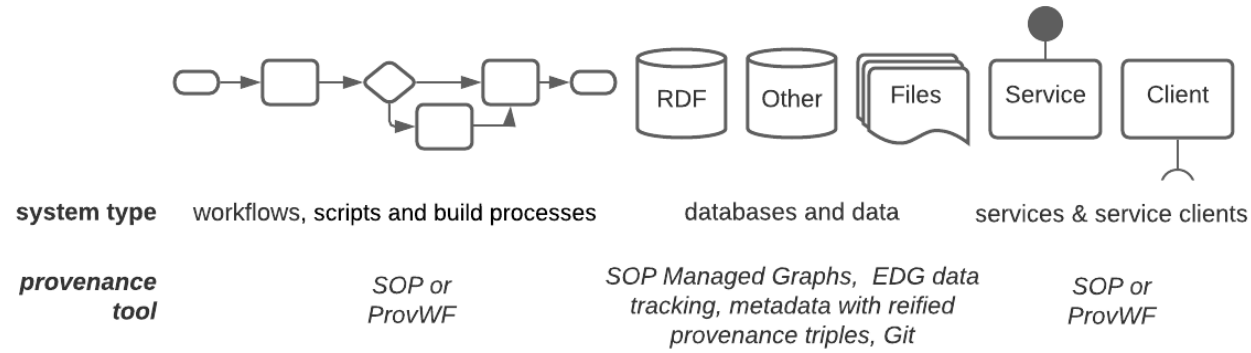

Figure 2: SURROUND's provenance tools linked to system type

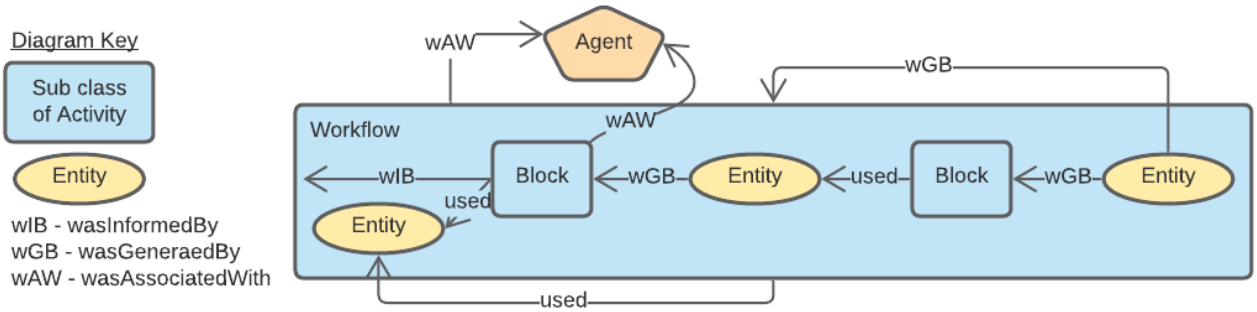

Figure 3: An example of the main PROV-O elements generated by *ProvWF*. Note that the tooling automatically captures identifiers for the versions of software used for any particular workflow implementatation

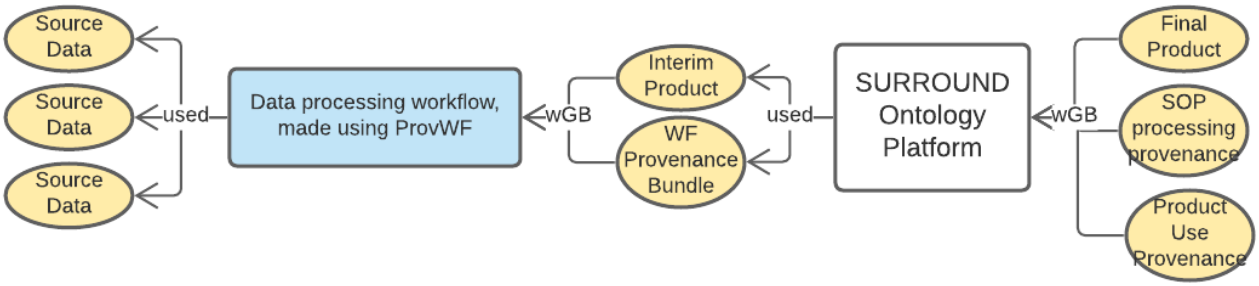

Figure 4: *ProvWF* is often used to generate RDF data - here the "Interim Product" - which can be supplied to *SOP* with an acompanying provenance `Bundle`. *SOP*, in turn, generates both `Bundles` of provenance for any actions on data it performs and also recurds usage provenance for products

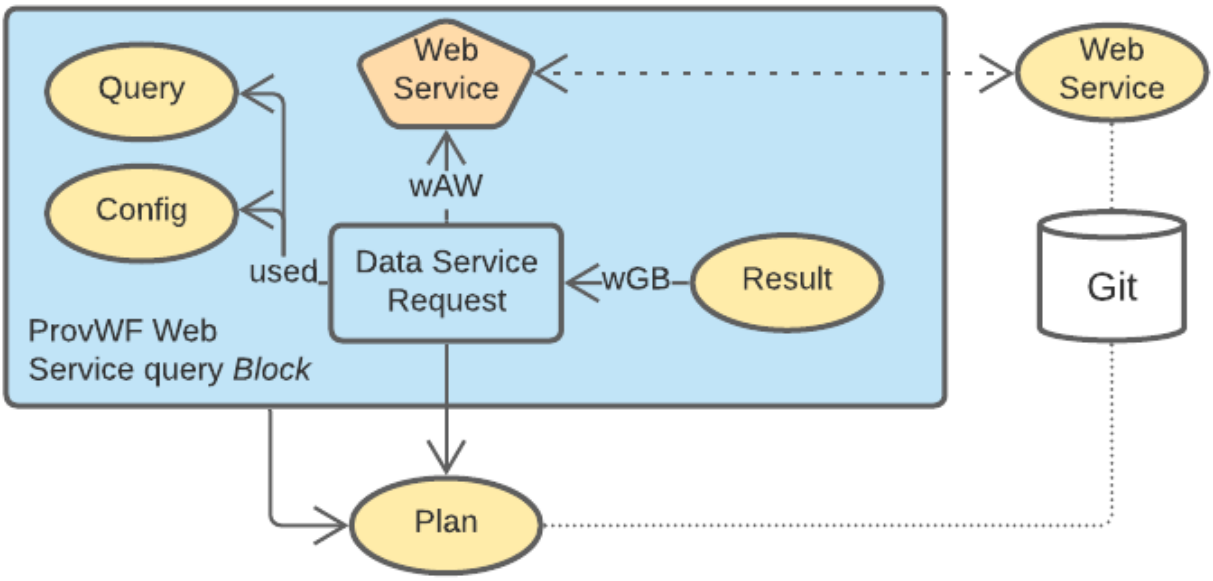

Figure 5: SURROUND's provenance tools linked to system type

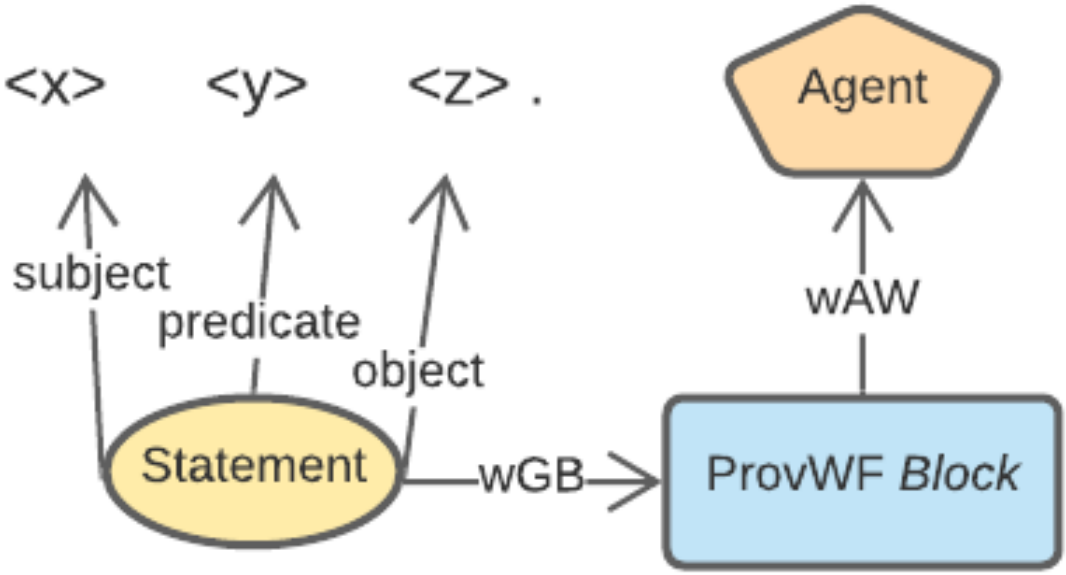

Figure 6: SURROUND's provenance tools linked to system type