An industrial provenance capability based on PROV for broad ranging IT projects including 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 & knowledge management. Company operations use standardised provenance tracking based on the PROV Data Model (PROV-DM). This allows us both assure customers that any results we produce for them are explainable, regardless of specific systems used, and also allows us to easily perform automated learning on top of about multi-part systems.

We generate PROV-DM provenance in important business workflows by using our *ProvWF* tool or workflows within our the *SURROUND Ontology Platform* (SOP) tool. We also implement PROV-DM within our project and company data which are mostly Knowledge Graph (KG)s. We "hand off" certain forms of provenance tracking - for some data and software - to Git-based systems but ensure that they integrate with our PROV-DM model.

1 Introduction

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

- the expressivity of RDFS[1] / OWL2[2] alows for infinitely complex, yet cohesive, data modelling
- the ability to directly reuse many existing, sophisticated, publishd models (ontologies)
- the extensibility of RDF graph-based data structures - no need to alter system schema as they grow
- the system-independence of Semantic Web data formats allowing for seamless under-the-hood system changes
- 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 [4]
- the advanced reasoning capabilities of OWL & SHACL to infer new knowledge

Emergent from some of these points is SURROUND's ability to provide comensurate provenance information across all our different systems.

In this paper we make no new research claim - this is an *Applications Track* paper - but we do claim to show "innovative use of provenance" and "the deployment of provenance-based solutions" that indicate a certain maturity of approach in 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, to help with the assessment of provenace research' impact and for researchers 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 provenance theory, complex practice

In general, extraction of useful information present within heterogeneous or large-scale data contexts may be performed in several ways. If some of the data has known structure then queries can be used to select relevant subsets. The trivial form

[1] https://www.w3.org/TR/rdf-schema/
[2] urlhttps://www.w3.org/TR/owl2-overview/

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 (KG)-based contextual information 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.

To record provenance systematically across our multiple systems and our various data processing domains requires a sensible provenance reference model and detailed system/data translation/communication/integration technical work. Since the advent and then widespread adoption of the PROV Data Model (PROV-DM) [6] and it's Semantic Web form PROV-O [5], we have a provenance reference model that SURROUND applies "everywhere" (as far as we practically can). The model is flexible enough for our use, with some small extensions, within our many systems and for many scenarios and cohesive enough for systematic use.

The technical implementation challenges we face are characterised as:

1. **Shared entity identification** - between different systems as they work on and pass the entities
2. **Granularity** - capturing useful levels of provenance detail while allowing for process or system level aggregations

The first we handle using the RDF, which uses unique URIs for things (entities and others) and the Open World Assumption: multiple systems can represent knowledge in separate RDF datasets and join them by referencing shared URIs. SURROUND not only uses RDf for provenance but, for most projects, primary data also. When project data, perhaps electronic records' RDF metadata, and provenance information for it are both represented in RDF, we can identify entities across both information holdings too, not just across sytems within each holding. For non-RDF information, such as Git-based software and data version information, we ensure that URIs are also used and referenced.

Our project data and provenance can be integrated across multiple subsystems if:

1. Object identity is established in KGs and they are accessed via APIs
2. Object identity is managed within the KG whenever HITL interactions are required
3. Processing subsystems preserve and report canonical object identities
4. Coherent sets of objects may be managed in specialised persistence systems, such as Git, as long as the description of datasets containing them are managed within KGs (see dataset granularity)
5. All processing reports' provenance, along with outputs, use our canonical model - PROV-DM
6. Processing elements are identified coherently in KGs

To achieve the above points, we have implemented systems and methods described in the next section. To demonstrate what such implementation allows for, we include Figure 1 which 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 (PROV) by SURROUND's processing workflow tool *ProvWF* performing Named Entity Recognition against electronic records matching entities against some of SURROUND's Knowledge Graph products. In addition to displaying the provenance information in particular ways, SOP also managed it in bundles as *Managed Graphs* which, from SOP's point of view are yet another semantic asset for which provenance (and ownership, access etc.) is automatically stored.

The provenance granularity issue can be addressed by examining a range of different types of processing that may typically occur in a heterogeneous systems and do occur within our projects. Table 1 lists processing functions, examples of them and (our) required granularity. We perform a per-scenario assessment of the required provenance granlarity for projects and then, since we are using PROV everywhere, usually employ existing tools of ours to create and store it.

3 Company-wide provenance architecture

Figure 2 links types of IT project assets we work with to the tools we implement to record PROV-O-compliant provenance. Our major provenance tools and the actions they perform are:

- **SURROUND Ontology Platform** (SOP)[3]
  - an enterprise data management system based on sematic data that is based on Top Quadrant's *EDG*[4]
  - SOP extends EDG adding management of semantic asset state and collections of them
  - SOP records PROV-DM provenance for all semantic asset actions

[3] https://surroundaustralia.com/sop

[4] https://www.topquadrant.com/products/topbraid-enterprise-data-governance/

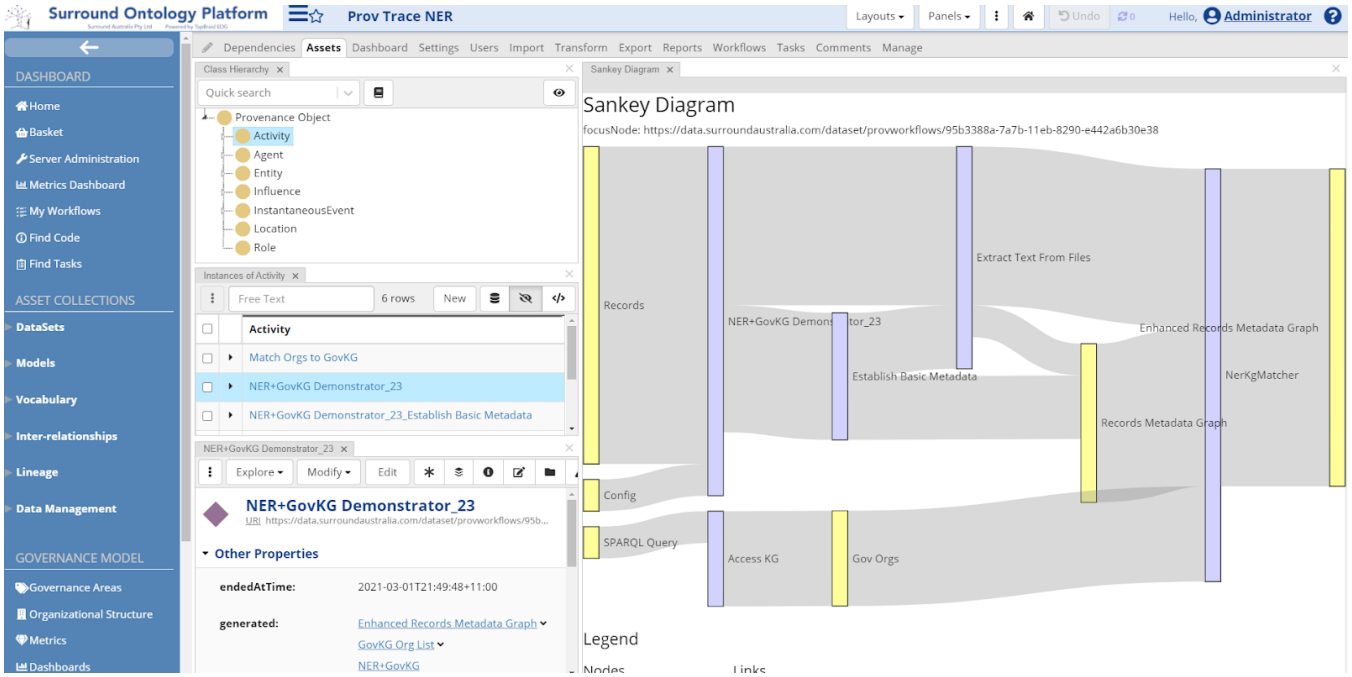

Figure 1: An example of a provenance trace from a processing workflow that uses elements of a knowledge graph, performs processing in cloud-hosted scalable services, generates augmented views of an input stream (performing Named Entity Recognition on a document set and annotating with elements from the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

- **ProvWorkflow** (ProvWF)[5]
  - Python framework for creation of workflows
  - records PROV-DM provenance for actions performed by the workflow and data consumed/produced
  - supported by SURROUND's *Block Library*
  - itegrates with SOP by provenance bundle transfer, see Figure 3
- **Block Library**
  - SURROUND's catalogue of ProvWF *Blocks*: PROV-DM `Activity` class objects
  - library stores many reusable functions within *Blocks*, such as KG API requests, NLP text processing etc.
- **Git**[6]
  - to track the versions of many assets - code, data etc. - in both public and private repositories
  - we have not yet found it necissary to materialise Git-to-PROV mappings, e.g. Git2PROV [8], but instead record URIs for entities managed in Git and refer to them in PROV-O data

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

- **RDFlib**[7]
  - general-purpose Python RDF manipulation library
  - many of our data objects are RDF graphs
  - we maintain RDFlib code blocks, e.g. to create reified provenance for RDF statements

4 Aspects of our provenance modelling

Much of our provenance modelling will be familiar to PROV users: chains of *Activities* and *Entities* associated with *Agents*. We do have several project scenarios which cause us to implement slight specialisations and two are given in project-specific case studies next.

[5] https://surroundaustralia.com/provwf
[6] https://git-scm.com/
[7] https://github.com/RDFLib/rdflib

Function	Examples	Granularity
Human-in-the-loop ML classification	Establishment of defs, Registration entities, Annotation, Classification for training	Statement, Reified statements
Database management, Data transformation	Making data instances sets available in a useful form	Dataset (table, spreadsheet, graph etc)
Query	Extraction of data subsets	Dataset, Resultset
Governance	Selecting particular datasets for use	Dataset
Bulk object processing	Indexing, classification, clustering	Whole-of-workflow
Document analysis	Making information elements in a document available to finer grained processes	Document, derived dataset
KG Management	Est'ment of state of complex, modular KGs, change tracking, support for automated updates	Graph (Dataset)

Table 1: A list of project functions, examples of them and (our) required provenance granularity

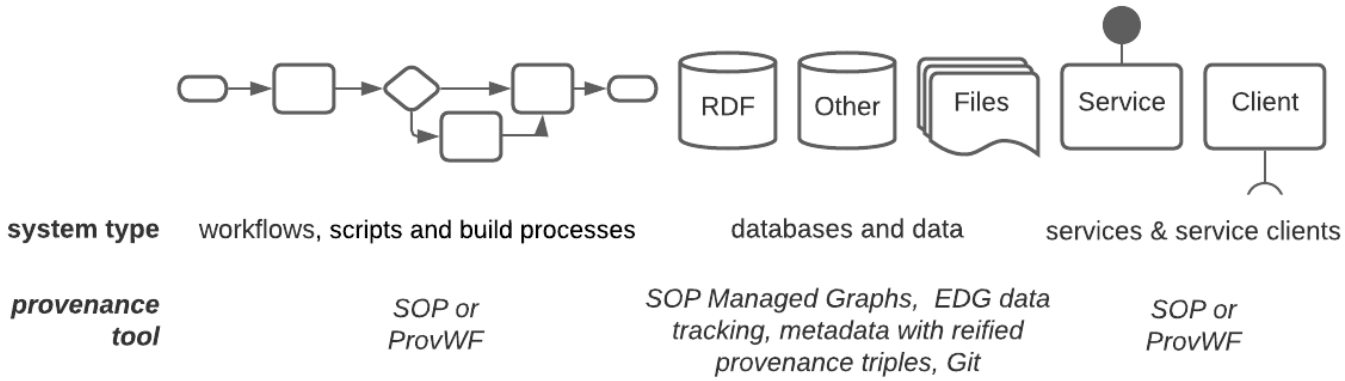

Figure 2: SURROUND's provenance tools linked to system type

4.1 Project 1: Electronic Records assessment

A recent project of ours has applied Natural Language Processing and KGs to electronic Records classification for archiving: we have extracted elements of Records' content, compared them with managed sources of *context* presented as KGs and used ML to learn how best to classify the Records.

Important in this project's provenance capture has been the use of data services - our classification workflow system querying our own KG services - and the tracking of both whole-workflow and individual data statement provenance. Figure 4 shows our model for data service query tracking within a workflow. Whole-of-workflow provenance is needed to track what configurations produce what results. All of the instances of a workflow's execution and the individual elements within it are recorded as PROV `Activity` instances and data as PROV `Entities`. Individual data statement provenance is needed for classifications within a Record's medata in RDF so individual Record results can be verified. Workflow, metadata and metadata provenance are all stored in an RDF database and are crosslinked.

Since shortly after the publication of PROV, SURROUND has watched extensions to it, or specialisations of it, for workflows. Our workflow modelling follows fairly simple PROV extensions such as PROV-Wf [3] however, we find that the `Plan`, or instructions for workflow execution are naturally recorded within the workflow's defining software code, so our *ProvWF* tool records a URI reference to the version of the code (the Git *commit* or *release*) that was executed. *ProvWF* requires custom PROV-style logging to be defined per custom *Block*, unless pre-defined *Block* instances from our *Block Library* are used. We have considered systems that generate PROV-compatable workflow provenance more automatically from defined workflow elements, such as [7] but have, so far, found that the level of workflow specification needed for those approaches (using a specialised business process modelling language) exceeds the provenance granularity we need for our workflows and forces us to simple define non-executable data structures as complex as simple directly defining PROV

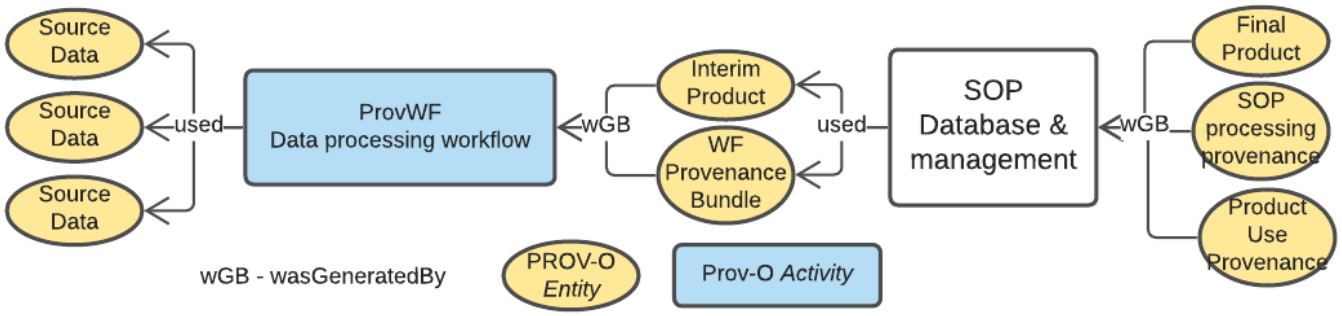

Figure 3: *ProvWF* is often used to generate RDF data - here the "Interim Product" - which can be supplied to *SOP* with an acompanying provenance `Bundle`. *SOP*, in turn, generates both `Bundles` of provenance for any actions on data it performs and also records usage provenance for products

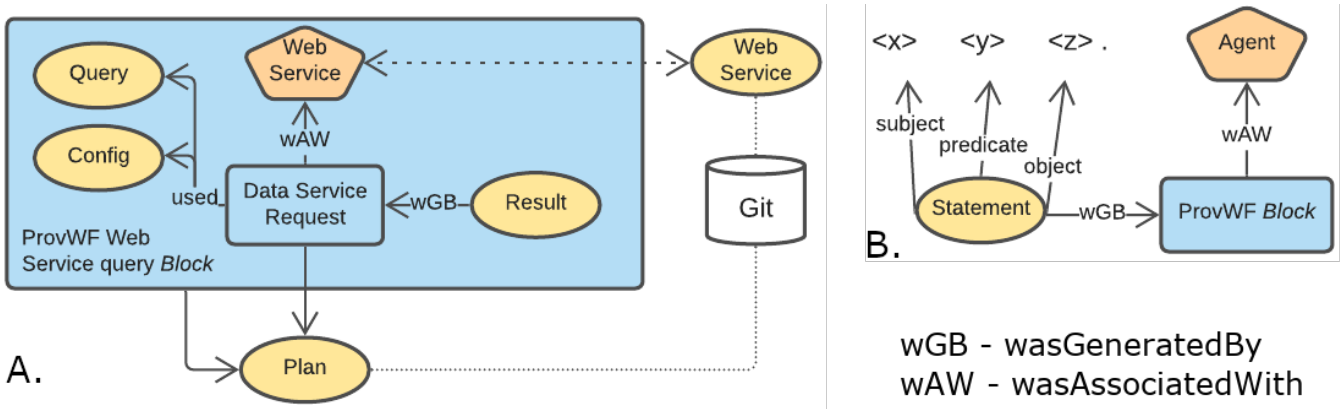

Figure 4: **A**. A *Service query block* from our *Block Library*, implemented within our *ProvWF* framework using a Query and other configuration (Config) to query a Web Service agent for a Result. Provenance for the web service itself, now considered an entity, is recorded in Git systems and referenced by each *ProvWF* execution. **B**. Reified provenance for a single RDF triple associated with the *ProvWF* Block instance that generated it.

itself (perhaps more so).

Some recent work on scientific workflow provenance [1] focus on modelling control flow to answer questions such as "what are the reasons for divergent results in two executions of a workflow?". Currently we do not implement such modelling in *ProvWF* or any other tool and instead represent workflows only as a simple *Workflow* containing *Blocks* that are arranged liniarly in time: all control flow decision making is subsumed into *Blocks* which does make them complex, but, so far, we have been able to model all interesting control flow choices as *Block* inputs or outputs. If, in future projects, we are interested to further focus on particular workflows' control flow elements, we anticipate representing them as a specialised *Block* with templated (expected) inputs and outputs and then comparing instances of that specialised *Block* to discover control flow choices.

4.2 Project 2: Document semantic querying

Another project of ours involved decomposing a large industry specification document into not only structural but also semantic elements (phrase meanings, synonyms, diagram descriptions, terminology lists, algorithm elements) to facilitate natural language and naïve searching of it. To track the efficacy of different document content decomposition methods and of different reference datasets - vocabularies of industry-specific terms etc. - we implemented a multi-system proveance regime. This involved modelling datasets (often KGs) as PROV *Entity* instances and tracking their state over time as queries were put to the multi-part system. We tracked the inbound queries to our system as per work some of our authors conducted many years ago [2] which treats each request as a PROV `Activity` from an anonymous `Agent` and for which results v. KG state are extracted from web logs.

Tracking reference dataset state in this project was done through the use of our SOP tool whcih has had graph state tracking capability added to it over many years and which allows for provenance about changes, new data insertions etc. to be recorded for both whole datasets within a set and also elements within the datasets. In SOP we can refer to the state of a collection of assets with a single URI reference - a *version* of an asset collection - and the provenance of queries and workflows quote that URI to ensure provenance cross querying is possible.

5 Reflections on PROV modelling

We have appreciated the graph-based nature of PROV-DM (in the PROV-O form that we use it) and find this aspec of PROV, along with the RDF methods for object identification and extensible schemas, to be the most useful aspect of PROV. These alow us to model "anything" at any level of granularity, store provenance data in one kind of system and cross query it to dive into elements, aggregate values etc.

Few aspects of PROV have been problematic for us but those that have include:

1. difficulty in storing a complex data in provenance graphs
    - when we wish to store complex objects for provenance but don't want to persist them in data store seperately to provenance KGs, we have struggled to think of how to represent them, without resorting to excruciating Semantic Web modelling
2. linking `Entity` instances to `Plane` instances
    - we want to be able to associate PROV `Entity` instances to the `Plan` instances used to instruct the workflows that generated them. PROV struggles to allow persstent `Entity` / `Plan` relations

6 Conclusions

SURROUND uses fairly standard patterns with PROV to model provenance across multiple systems and within different IT domains. We have easily adapted the PROV model to provide provenance that gives our customers confidence in the results they receive from us via access to individual results' lineage, particularly important for complex AI/ML applications, and gives us the ability to learn how our multi-part systems perform against tasks through cohesive logging. We have put considerable effort into establishing technical interfaces, interactions and integrations between our tools. We have developed both dedicated provenance tools and provenance capability within generic tools. We have not, so far, had to use highly specialised versions of PROV to achieve our goals.

References

[1] Anila Sahar Butt and Peter Fitch. A provenance model for control-flow driven scientific workflows. *Data & Knowledge Engineering*, page 101877, February 2021.

[2] Nicholas John Car, Laura S Stanford, and Aaron Sedgmen. Enabling Web Service Request Citation by Provenance Information. In Marta Mattoso and Boris Glavic, editors, *Provenance and Annotation of Data and Processes: 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Proceedings*, pages 122–133. Springer International Publishing, Cham, 2016.

[3] Flavio Costa, Vítor Silva, Daniel de Oliveira, Kary Ocaña, Eduardo Ogasawara, Jonas Dias, and Marta Mattoso. Capturing and querying workflow runtime provenance with prov: A practical approach. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, page 282–289, New York, NY, USA, 2013. Association for Computing Machinery.

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

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

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

[7] Ajinkya Prabhune, Aaron Zweig, Rainer Stotzka, Michael Gertz, and Juergen Hesser. Prov2ONE: An Algorithm for Automatically Constructing ProvONE Provenance Graphs. In Marta Mattoso and Boris Glavic, editors, *Provenance and Annotation of Data and Processes*, Lecture Notes in Computer Science, pages 204–208, Cham, 2016. Springer International Publishing.

[8] 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.