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 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.

Figure 5 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 5, 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:

- **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.

3 Footnotes, Verbatim, and Citations

Footnotes should be places after punctuation characters, without any spaces between said characters and footnotes, like so.[6] And some embedded literal code may look as follows.

```
int main(int argc, char *argv[])
{
    return 0;
}
```

Now we're going to cite somebody. Watch for the cite tag. Here it comes. Arpachi-Dusseau and Arpachi-Dusseau co-authored an excellent OS book, which is also really funny, and Waldspurger got into the SIGOPS hall-of-fame due to his seminal paper about resource management in the ESX hypervisor.

The tilde character (˜) in the tex source means a non-breaking space. This way, your reference will always be attached to the word that preceded it, instead of going to the next line.

And the 'cite' package sorts your citations by their numerical order of the corresponding references at the end of the paper, ridding you from the need to notice that, e.g, "Waldspurger" appears after "Arpachi-Dusseau" when sorting references alphabetically.

It'd be nice and thoughtful of you to include a suitable link in each and every bibtex entry that you use in your submission, to allow reviewers (and other readers) to easily get to the cited work, as is done in all entries found in the References section of this document.

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

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

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

[4] https://surroundaustralia.com/provwf

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

[6] Remember that USENIX format stopped using endnotes and is now using regular footnotes.

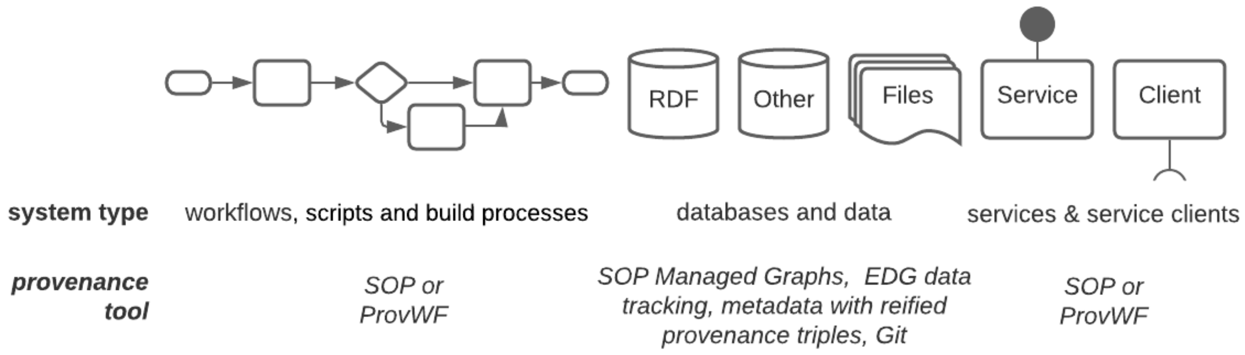
RDF
Other
Files
Service
Client
system type
workflows, scripts and build processes
databases and data
services & service clients
provenance tool
SOP or ProvWF
SOP Managed Graphs, EDG data tracking, metadata with reified provenance triples, Git
SOP or ProvWF

Figure 1: SURROUND's provenance tools linked to system type

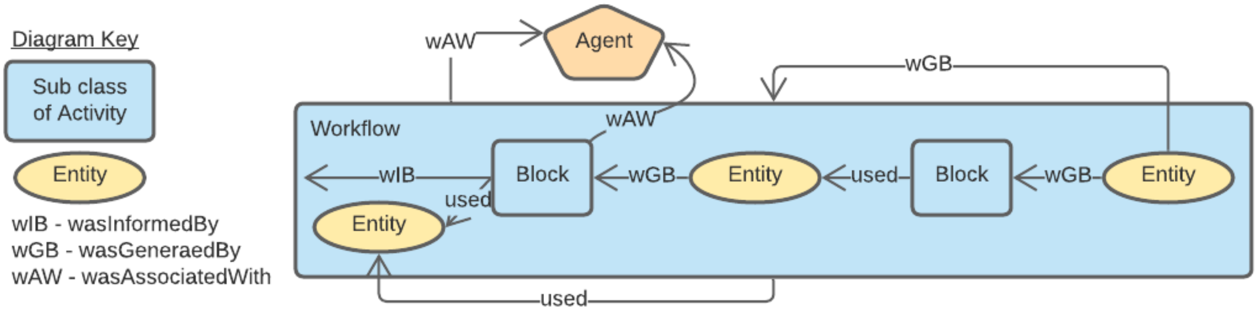

Figure 2: An example of the main PROV-O elements generated by *ProvWF*. Note that the tooling automatically captures identifiers for the versions of software used for any particular workflow implementatation

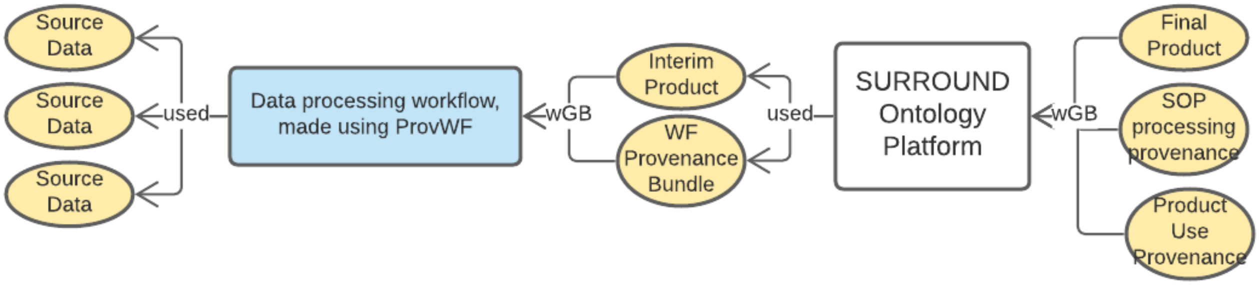

Figure 3: *ProvWF* is often used to generate RDF data - here the "Interim Product" - which can be supplied to *SOP* with an acompanying provenance `Bundle`. *SOP*, in turn, generates both `Bundles` of provenance for any actions on data it performs and also recurds usage provenance for products

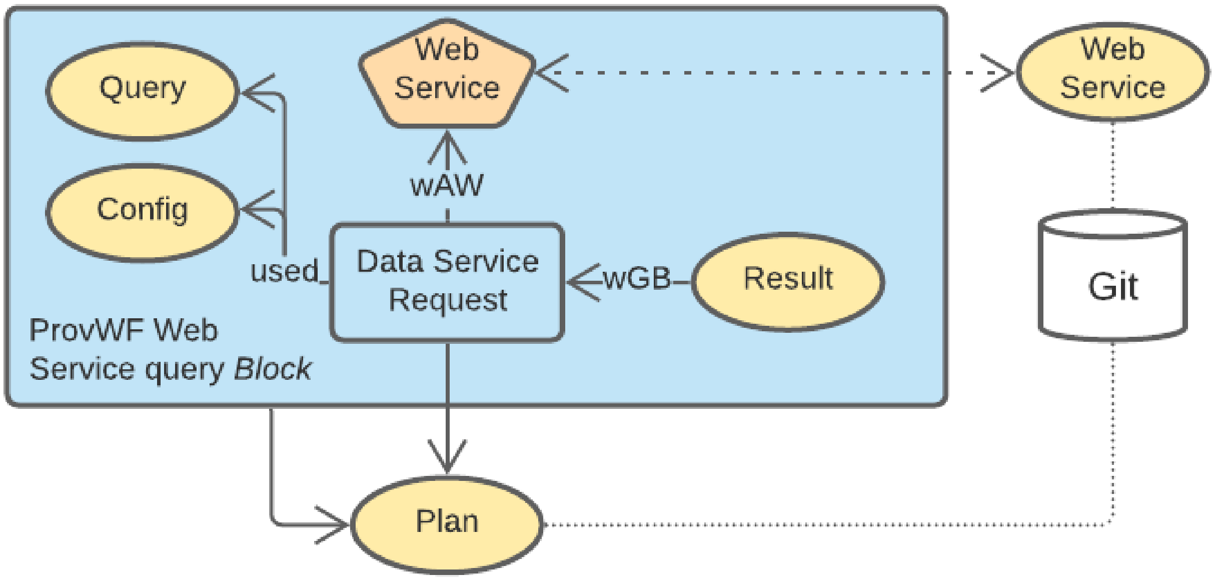

Figure 4: SURROUND's provenance tools linked to system type

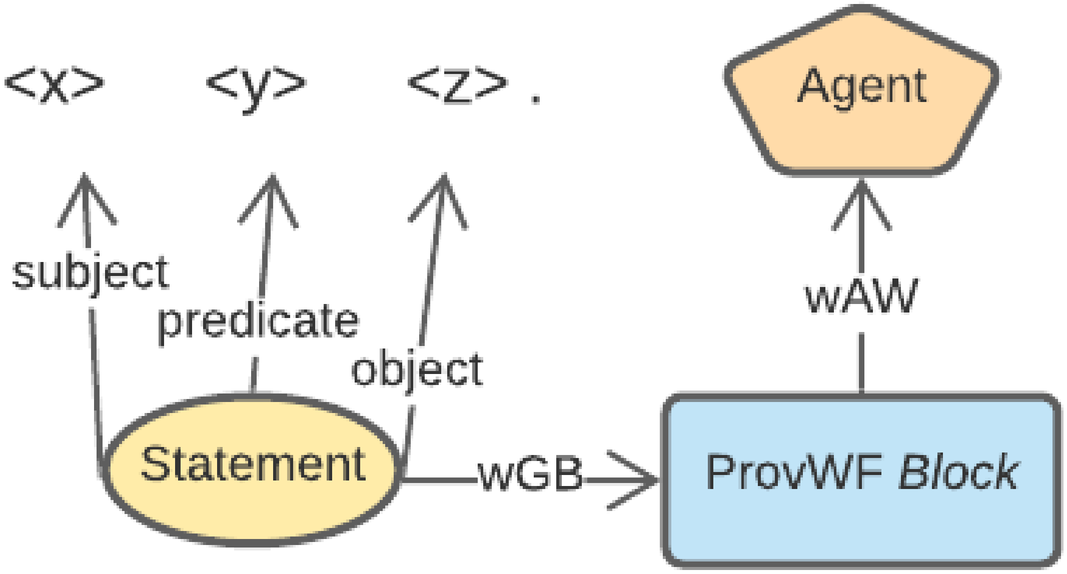

Figure 5: SURROUND's provenance tools linked to system type

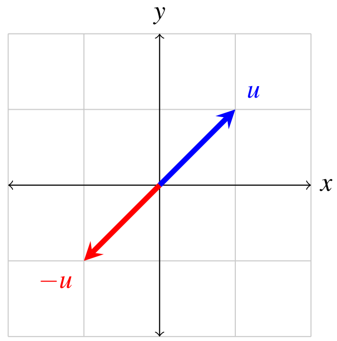

Figure 6: Text size inside figure should be as big as caption's text. Text size inside figure should be as big as caption's text. Text size inside figure should be as big as caption's text. Text size inside figure should be as big as caption's text. Text size inside figure should be as big as caption's text.

Now we're going take a look at Section 4, but not before observing that refs to sections and citations and such are colored and clickable in the PDF because of the packages we've included.

4 Floating Figures and Lists

Here's a typical reference to a floating figure: Figure 6. Floats should usually be placed where latex wants then. Figure6 is centered, and has a caption that instructs you to make sure that the size of the text within the figures that you use is as big as (or bigger than) the size of the text in the caption of the figures. Please do. Really.

In our case, we've explicitly drawn the figure inlined in latex, to allow this tex file to cleanly compile. But usually, your figures will reside in some file.pdf, and you'd include them in your document with, say, \includegraphics.

Lists are sometimes quite handy. If you want to itemize things, feel free:

fread a function that reads from a `stream` into the array `ptr` at most `nobj` objects of size `size`, returning returns the number of objects read.

Fred a person's name, e.g., there once was a dude named Fred who separated usenix.sty from this file to allow for easy inclusion.

The noindent at the start of this paragraph in its tex version makes it clear that it's a continuation of the preceding paragraph, as opposed to a new paragraph in its own right.

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.