Facility design metadata as Rdf

Dag Hovland, Eirik Nordstrand

We describe certain data-engineering issues that arise when trying to go from a document-centric spreadsheet-based representation into a model-centric Rdf-based representation of early design information on oil platforms. The content of the documents usually have well-defined stakeholders and standards in place. We describe a solution to handling the metadata of these documents.

Problem description *Master Equipment Lists* are documents that contain very early design information about weight and certain other properties of a new oil or gas facility. This information is usually transmitted in large excel spreadsheets. Our overall goal is to get away from the restrictions imposed by using spreadsheets and leverage the possibilities of a graph-based format like Rdf. In this paper we will not be concerned with the representation of the engineering design data, but rather about the information contained in the *envelopes* of the data. This includes filename, transfer date, sender, contract number etc. We will call all this data *provenance*, while the data describing the facility itself will be called *content*. So our aim is to find efficient, precise, and preferrably reusable data models for this data.

Data Model: Records The core of our approach is the reification of the partial versions of the actual *content* into immutable named graphs that we call *records*. The records are identified by the IRI of the named graph, and in addition to the content, the records also contain triples that have the named graph IRI as subject. These triples are the provenance of the record. It is easy to detect the difference, since the record is typed with a dedicated class. The content in a named record is by agreement/specification not allowed to change. The provenance in a record is append-only. Records are defined and described in https://github.com/equinor/records.

A record can be related to one or more other records with a `replaces` relation. These other records represent previous, now presumably outdated (or *replaced*), information. The directed acyclic graph spanned out by `replaces` represents the history of the data. The records that are not replaced are collectively called the "head". End-user applications of records are expected to usually work on the union of the content in the records in the head.

Records also have a relation `describes` to resources that are present in the content and `isInScope` to scope the description of those resources.

ISWC 2023 Industry Proceedings
EMAIL: dag.hovland@bouvet.no (D. Hovland); eirik.nordstrand@webstep.no (E. Nordstrand)
ORCID: 0000-0002-3569-8838 (D. Hovland)



CEUR Workshop Proceedings (CEUR-WS.org)

The intuition is that the content in a record is a description of the elements it `describes`, but limited to the scope in `isInScope`. In the head, for any given set of scopes, the relation `describes` is injective for the set of records that have all those scopes. Or, in other words, there is an injective relation that takes pairs of a set (of scopes) and an element (`describes`) into the set of records that are not replaced.

That is, there cannot be two records in the head that have a `describes` relation to the same thing. However, the `describes` relation is not restricted across `replaces`, so objects can change name across these relations. Also, objects can be "split up", in the sense that multiple records can be related with `replaces` to a single record as long as the restriction on non-overlapping `describes` is kept in the new head. Splitting up records in this way is useful to smooth the transition from document-centric to model-centric, since the documents are often describing many objects.

Use of the data model: Revisions A major concern among end users when going from document-centric to model-centric was to lose the metadata related to the spreadsheets. Records make it easy to attach provenance to certain versions of data, and to track that provenance over time. For our use case, we created two new resources / IRIs: The document as a persistent object over time and changes, and the revision as an unchangable collection of records, representing a specific version of the document

The revision has a relation to the document it is a revision of, and relations to the records that contain the content in that speicific version. In this way, long-lived information like contracts, facilities, etc. can be put on the document object, while information about specific versions, specifically, comments and approvals, can be made on the revisions. It is also possible to compare the information in different revisions of the same document by comparing the union of the content in the respective sets of records.

Conclusion Other approaches to versioning Rdf have been described by Auer [1], Ma [2], and Hovland et al. [3]. We conclude that the techniques described above are new and make it easier to introduce Rdf to represent facility design data.

References

[1] S. Auer, H. Herre, A versioning and evolution framework for RDF knowledge bases, in: I. B. Virbitskaite, A. Voronkov (Eds.), Perspectives of Systems Informatics PSI 2006, volume 4378 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 55–69. doi:10.1007/978-3-540-70881-0_8.

[2] X. Ma, C. Ma, C. Wang, A new structure for representing and tracking version information in a deep time knowledge graph, Computers & Geosciences 145 (2020) 104620. doi:https://doi.org/10.1016/j.cageo.2020.104620.

[3] D. Hovland, F. Chrislock, Versioned objects, in: A. Dimou, A. Haller, A. L. Gentile, P. Ristoski (Eds.), Proceedings of the ISWC 2022 Posters, Demos and Industry Tracks, CEUR Workshop Proceedings, 2022. URL: https://ceur-ws.org/Vol-3254/paper398.pdf.