Managing the Euclid Data Model

Nutma, T. A.; Dabin, C.; Valentijn, E. A.

Published in:
Astronomical Data Analysis Software and Systems XXIX

Astronomical Data Analysis Software and Systems XXIX
ASP Conference Series, Vol. 527
R. Pizzo, E.R. Deul, J-D. Mol, J. de Plaa and H. Verkouter, eds.

Managing the Euclid Data Model

Teake Nutma,[1] Christophe Dabin,[2] and Edwin A. Valentijn[1]

[1]*Kapteyn Astronomical Institute, University of Groningen, The Netherlands;* t.a.nutma@rug.nl

[2]*Centre National d'Études Spatiales, Toulouse, France*

Abstract. The Euclid common data model is central in, and essential to, the Euclid science ground segment. It defines the format of all data exchanged between the pipelines and stored in the Euclid Archive, and ensures all components can communicate with each other. But with more than 25 active contributors, managing the data model has been a challenge. Care must be taken that changes in the XML of the data model do not break its Python, C++, or database bindings.

We describe recent progress in tackling these problems. The former problem has been mitigated with a new data model validator tool run during continuous integration. The latter has partially been solved via git management rules. Both approaches have only been possible after the migration of SVN to git, allowing the introduction of modern tooling.

1. Introduction

Euclid is an ESA medium class mission aimed to better understand the nature of dark energy and dark matter (Laureijs et al. 2011). It is slated to launch in 2022, and will measure photometric and spectroscopic redshifts of galaxies. Once at its nominal L2 orbit it will cover 15000 deg^2 in a wide survey and 40 deg^2 in a deep survey, two orders of magnitude deeper than the wide survey.

The completed survey, combined with ground-based surveys in order to achieve the scientific objectives, will comprise hundreds of thousands images, several tens of petabytes of data, and about ten billion sources. How this data is structured, handled during processing, and stored in the Euclid archive (Nieto et al. 2019), is all provisioned in the Euclid Common Data Model.

2. The Euclid Common Data Model

The Euclid Common Data Model (ECDM) is mainly written in XML Schema Definition (XSD) files. It encodes the structure of some 183 data products (ranging from raw LE1 telemetry to science ready data products), 20 system interfaces, and 200 FITS file

formats. Given this all-encompassing task, it is rather complex:[1] the ECDM has more than 1000 definitions and 3000 dependencies between them (see Figure 1).

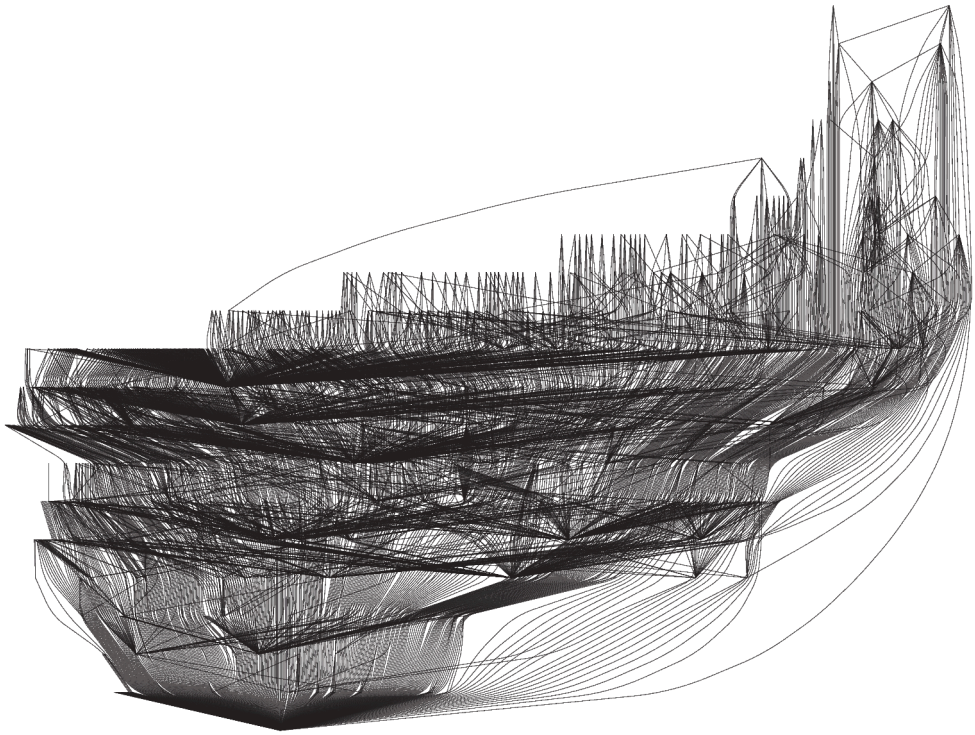

Figure 1. A graph representation of the ECDM. Points represent XSD type definitions, and lines dependencies between them. Complex composite types (e.g. science ready data products) are on the top of the graph, whereas simple type (e.g. strings, integers, and floats) are on the bottom.

The XSD files are used to validate the structure of XML metadata and FITS data passed between the Science Ground Segment components during processing (Frailis et al. 2019). These components include e.g. the archive, an Infrastructure Abstraction Layer, and distributed HPC clusters.

Moreover, the data model XSD is also used to generate XML bindings to Python and C++, for both the archive and the processing pipelines running on the HPC clusters. In case of the archive, the Python bindings also serve as a Object-Relational Mapping (ORM) to a relational database. Thus the schema of the archive database can be mapped one-to-one to the XSD of the data model.

[1]Euclid has two data models, the ECDM and the Science Exploitation Data Model. The latter is geared towards science consumption and is a subset of the full ECDM, and as a consequence much simpler.

3. XML Rules

However, this presents a problem. The W3C specification of XSD allows far more freedom than is permitted in a relational database schema or in Python or C++ code. To give a concrete example, consider the representation of integers. XSD has both unbounded types (e.g. `xs:integer`) and bounded types (e.g. `xs:int` for a 32-bit signed integer), whereas in a database you typically want to use only bounded numerical types. Another restriction is that in Python you cannot use reserved keywords as package names. This in turn implies that XSD namespaces should not contain a Python reserved keyword, although this is perfectly legal from the W3C specification.

All such domain-specific rules are collected in a reference document and implemented in a command line validator tool. These so-called "XML-rules" simplify readability, re-usability, and the implementation of the data model. It is therefore crucial they are adhered to by all contributors to the data model. This is the reason why the validator tool was introduced; the reference document proved to be too impractical for every-day use, and a significant amount of errors crept into the data model (see Figure 2).

4. Automated Validation

The migration of the version control of the ECDM from SVN to git allowed the validator to be run for every commit, and enabled the introduction of strict management rules. The workflow for contributing to and changing the ECDM is now as follows:

1. A contributor creates a new feature branch from the main development branch.
2. Changes are made in the newly created feature branch.
3. The contributor creates a so-called Merge Request to ask for their changes to be merged back into the main development branch.
4. Custodians of the part of the ECDM that has been changed review the changes, and check whether or not the validator has run successfully during Continuous Integration (CI).
5. Only when the CI pipeline has passed, the feature branch can be merged back into the main development branch.

Because the Euclid git hosting platform makes it possible to only enable the merge button on a Merge Request when the CI pipeline has passed, it is nearly impossible to introduce errors once this feature has been enabled. In fact, the ECDM has remained error-free once strict validation was in place (see Figure 2)[2].

[2]The small bump in the beginning of March 2018 was due to a stale Merge Request which had been run with an outdated version of the validator. The then-current validator was more strict and caught more errors, but the CI pipeline was not run again directly before the Merge Request was merged.

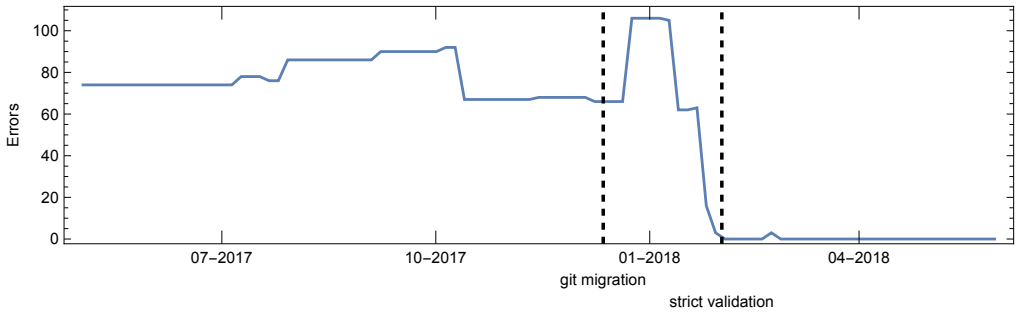

Figure 2. Errors in the ECDM over time. The sharp decline in errors was during a transition period, indicated with dashed lines, when automatic validation was enabled but not yet enforced. During this period errors were actively fixed. After all errors were fixed strict validation was enforced.

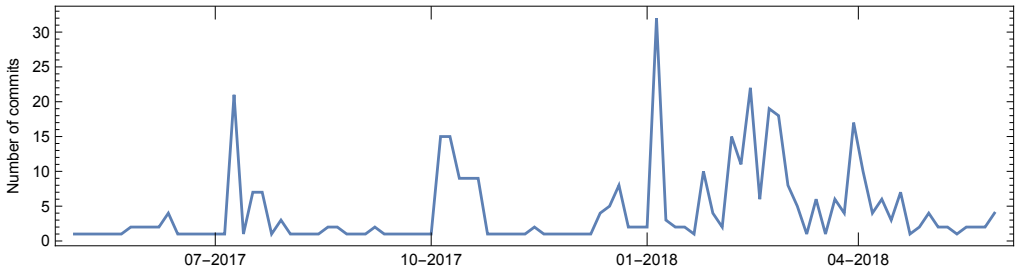

Figure 3. Commit activity for the ECDM over time.

5. Summary

The Euclid Common Data Model plays an integral role in the science ground segment, and is thus subject to a large set of domain-specific constraints. Having this set of constraints tested via Continuous Integration gives contributors immediate feedback on the validity of their changes. Enforcing the validation keeps the ECDM error free, and dramatically reduces the time and effort needed to integrate the various components of the ground segment.

References

Frailis, M., et al. 2019, in Astronomical Society of the Pacific Conference Series, edited by M. Molinaro, K. Shortridge, & F. Pasian, vol. 521 of Astronomical Society of the Pacific Conference Series, 612

Laureijs, R., et al. 2011, arXiv e-prints, arXiv:1110.3193. 1110.3193

Nieto, S., et al. 2019, in Astronomical Society of the Pacific Conference Series, edited by M. Molinaro, K. Shortridge, & F. Pasian, vol. 521 of Astronomical Society of the Pacific Conference Series, 12