DISCOS 3: AN IMPROVED DATA MODEL FOR ESA'S DATABASE AND INFORMATION SYSTEM CHARACTERISING OBJECTS IN SPACE

F. McLean[(1)], S. Lemmens[(2)], Q. Funke[(3)], and V. Braun[(3)]

[(1)]*CS GmbH at ESA/ESOC, Robert-Bosch-Str. 5, 64293 Darmstadt, Germany, Email: frazer.mclean@esa.int*
[(2)]*ESA/ESOC, Robert-Bosch-Str. 5, 64293 Darmstadt, Germany*
[(3)]*IMS Space Consultancy GmbH at ESA/ESOC, Robert-Bosch-Str. 5, 64293 Darmstadt, Germany*

ABSTRACT

DISCOS (Database and Information System Characterising Objects in Space) is an important resource for ESA's Space Debris Office. DISCOS maintains reference information for launches, launch vehicles, and spacecraft. This data includes physical attributes, mission objectives, and object registration details. Orbital data history, fragmentations, and re-entry events are stored for all objects, of which there are more than 45,000 entries.

DISCOS3's new data model is catalogue-independent, allowing storage of data from multiple sources. A flexible abstract object system allows the description of launch vehicles and spacecraft with relationships and encapsulation.

Rich object information can be stored, such as objects that dock to, contain, or release other objects; including a history of such changes. Fragments and debris from other objects may also be linked to their parent objects. Orbital data is linked to the catalogue-independent DISCOS object, providing a unified access layer for orbital states.

Keywords: DISCOS; debris.

1. INTRODUCTION

The European Space Agency (ESA) maintains DISCOS in support of its operational and academic activities. The system has been in operation since 1990 and is maintained by the Space Debris Office at the European Space Operations Centre (ESOC) in Darmstadt, Germany[1, 2].

Section 2 starts with an explanation of the DISCOS 2 object model and the changes in DISCOS 3. Changes to the record of launches and the correlation of launch vehicles and abstract objects are provided in section 3. Catalogue independent orbital data is discussed in section 4. Finally, changes to the owner/operator model are given in section 5.

2. OBJECT MODEL

DISCOS 2's object model used the COSPAR International Designator (COSPAR ID) and Satellite Catalogue Number (SATNO) to associate objects with related data, including orbital data, object properties, and launch information. The choice to use external catalogue identifiers as primary identifiers in DISCOS presented a major limitation; an object not present in those catalogues could not be represented in DISCOS in the same way. An example of this is rocket bodies that cleanly de-orbit and are therefore not catalogued. Additionally, should external identifiers or the mapping between them change—perhaps due to a transposition error or erroneous data—those identifiers had to be altered throughout DISCOS.

DISCOS 3 uses a surrogate key for objects. Optionally, external identifiers may be added for a given DISCOS object. These include the COSPAR ID, SATNO, and JSC Vimpel space object number (Vimpel ID). Additionally, the DISCOS mapping of identifiers is temporal. An epoch range can be assigned to a mapping from the DISCOS ID to external identifiers. This can be used when an on orbit collision results in the destruction of those objects. Table 1 gives an example for Iridium 33. The temporal identifier mapping allows the pre-collision DISCOS object to be distinct from the post-collision fragment that inherits its COSPAR ID. This allows the separate DISCOS objects to each have their own properties and metadata. Note that information about the existence or lifetime of the object is not included as the lower or upper bounds; to include this data here would prevent lookup of an object by ID after it no longer existed. It would also be a duplication of data and therefore allow inconsistencies.

2.1. Abstract objects

DISCOS 3 stores abstract objects—which in contrast to the objects referenced by DISCOS IDs—do not necessarily represent space objects[1]. Abstract objects are a type

[1] Specifically, objects that have a DISCOS ID but which are not necessarily in space.

or class of object, such as a specific model of satellite. Each DISCOS object references the abstract object that it is an instance of. For example, each Cluster II spacecraft (flight models 5 through 8) reference the same abstract object. In contrast to DISCOS 2, this prevents inconsistencies arising from duplication of data. DISCOS objects may also have properties independent from their abstract object. E.g. in the case above for Cluster II, the mass can be set for each spacecraft based on actual fuel consumption.

Abstract objects can be of arbitrary types. There is a base table with attributes common to all types, such as name and properties. Specific types of abstract object may have attributes unique to that type, such as maximum thrust for the engine type. Types of abstract object are implemented using a primary foreign key back to the base table. Abstract objects are used to represent the components of launch vehicles, which is discussed further in section 3.1.

2.2. Spatial Relationships

Spatial relationships between objects can be encoded in the DISCOS 3 path history. The path history starts at the first epoch for a given object, and ends when the object fragments, re-enters, or is returned. It is modelled using a tree. DISCOS objects may be contained within or connected to another. Additionally, these relationships are temporal. This allows a history of each object's path in the tree to be stored, and therefore multiple flight phases to be captured. Figure 1 shows an example model of the ISS with a subset of its modules. The relationship within the dashed outline exists only for the duration that the spacecraft is docked.

All the relationships in the previous example are "connected to" their parent, but DISCOS also allows "contained by" relationships. This could be used to model the path of a CubeSat. Initially, contained by its cargo vessel, then moved to the ISS and eventually released. At each phase of flight, the object catalogue can be reconciled to find the number of independent objects in space. Table 2 shows an example path history of the TuPOD CubeSat and one of the micro satellites it is carrying. At each example epoch, the full path of TuPOD and TANCREDO 1 is shown. After release from the Japanese Experiment Module (JEM), TuPOD is no longer marked 'contained', and that is reflected in its path. TANCREDO 1 remains contained within TuPOD until its release at the next epoch.

The container object in fig. 1 is not physical, but a recognition that a group of objects can be a single entity. Additionally, container objects in DISCOS can proxy the external identifiers for another object. In the example, the ISS container proxies the COSPAR ID and SATNO from Zarya to better reflect what those identifiers refer to.

DISCOS reconciles the spatial relationship tree differently for different use cases. It is possible to sum the mass from child components, for example. By filtering

DISCOS ID	COSPAR ID	Epoch range
1	1997-051C	$[-\infty, \text{2009-02-10 16:56})$
2	1997-051C	$[\text{2009-02-10 16:56}, +\infty)$

Table 1. An example of an identifier change for Iridium 33 post-collision.

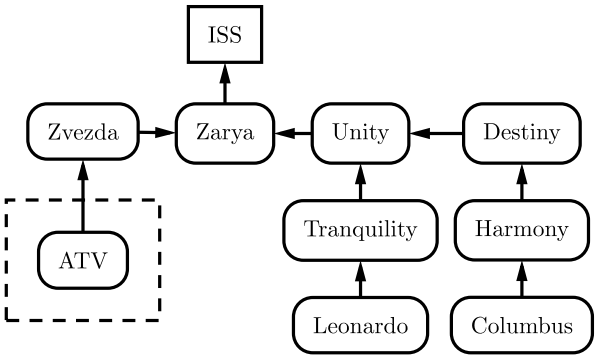

Figure 1. Spatial relationships for ISS modules, including a docked Automated Transfer Vehicle (ATV). Round rectangles are objects, rectangles are container objects.

Epoch	Path	State
00:00	JEM → TuPOD	contained
00:00	JEM → TuPOD → TANCREDO I	contained
01:00	TuPOD	–
01:00	TuPOD → TANCREDO I	contained
02:00	TuPOD	–
02:00	TANCREDO I	–

Table 2. An example of a CubeSat release from the ISS and subsequent "TubeSat" release.

the epoch range, it is possible to view the object catalogue at any time in the past. This makes it trivial to query the catalogue for a time series of mass evolution or number of objects. An example of this is given in fig. 2.

2.3. Re-entry Events

DISCOS 2 simply stored a re-entry epoch for each object. In order to provide additional data, such as location and whether it was observed, DISCOS 3 stores re-entry events separately. This also allows multiple objects that re-enter together (as with the new spatial relationships) to reference the same re-entry event.

2.4. Fragmentation Events

In DISCOS 2, fragmentations were recorded per object. While this is the common case, this did not allow multiple-object fragmentations to be modelled. DISCOS 3 uses fragmentation events and many-to-many relationships to the relevant DISCOS objects.

Within the DISCOS 3 data model, objects created from a fragmentation are independent. In order to determine a fragment's parent object, the path history table references that parent. This relationship can not be modelled using the previously mentioned spatial relationships, because the fragment did not exist pre-collision. It would be inadequate to model fragments retroactively as connected to their parents since launch.

For pragmatic reasons, mission-related objects in DISCOS are treated in the same way, with a reference to the parent in the path history table.

2.5. Object Names

DISCOS 2 kept a single ASCII name per object. It is often the case that an object is known by multiple names. DISCOS 3 allows Unicode everywhere. Objects can also have additional names associated, including original names in other alphabets.

3. LAUNCHES

DISCOS 2's record of launches used the year and launch number of the COSPAR ID. Launches without these designations could not be recorded, including the launch of ESA's Intermediate eXperimental Vehicle (IXV) on 11 February 2015. The IXV launch, launch failures, and other launches without COSPAR IDs had to be recorded in the schema separately.

DISCOS 3 uses a surrogate key for launches, much like DISCOS objects. The year and launch number of the COSPAR ID are stored as a mapping of external identifiers to the DISCOS launch ID. Launch failure is indicated by a boolean flag and a text field for additional information.

3.1. Launch Vehicles

Launches reference the relevant launch vehicle. In DISCOS 2, launch vehicles were kept in a fixed table structure. The separate launch vehicle tables meant that if there was a DISCOS object corresponding to a part of a launch vehicle, that relationship could not be specified. This information allows simple queries to be made, such as finding which launch vehicles have upper stages present in the object catalogue.

DISCOS 3 uses abstract objects to store launch vehicles and their components. There are several abstract object types used for this purpose, including 'launch vehicle', 'configured stage', 'stage', and 'engine'. The 'configured stage' is a pseudo-type which only has a stage number attribute that should apply to its 'stage' child in the node tree. The node tree brings a collection of abstract objects together to form a launch vehicle.

For launches, the root abstract object is of the 'launch vehicle' type. An example of the structure is shown in fig. 3. The configured stage type exists because it is possible for a stage to be used as a different stage number in different launch vehicles. That is to say, the stage number is a property of the launch vehicle rather than the stage itself.

In the database, the node tree is a hierarchy of foreign keys to abstract objects. Each node also has a quantity field, used when there is more than one of a specific engine or stage in a launch vehicle. The node tree must be separate from the abstract objects, or an abstract object could not be present in more than one tree.

- Launch vehicle
  1. Configured stage 1
     - Stage
       - Engine
  2. Configured stage 2
     - Stage
       - Engine
  3. Configured stage 3
     - Stage
       - Engine

Figure 3. Node tree for abstract objects that represent a launch vehicle.

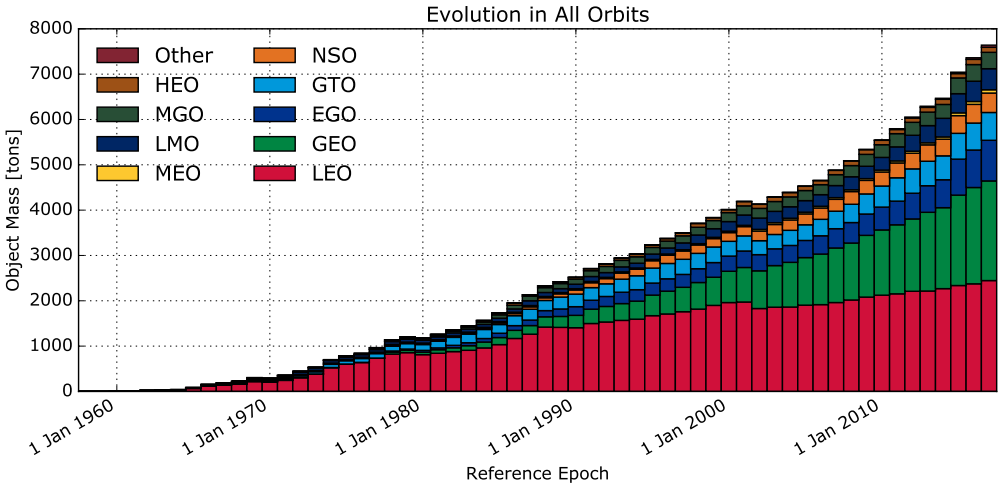

Figure 2. Mass evolution by orbit type.

4. ORBITAL DATA

As discussed in section 2, DISCOS objects use a surrogate key. Orbital data from external catalogues and ESA internally have their own identifiers depending on the provider. Source data from these catalogues is ingested into the database automatically. These data catalogues are used to source standardised orbit tables using the same external identifiers. The standardised form includes the Cartesian state, the reference frame, a reference to the source data, and optionally osculating elements. The orbit tables are sparse compared to the source data. They are pruned and updated when new data is ingested.

The external identifiers from the standardised orbit tables are reconciled with a DISCOS ID in a view, providing catalogue-independent access to orbital states. States keep their frame and catalogue of origin, should filtering be necessary for a given analysis.

4.1. Analyst Orbits

DISCOS 3 uses an analyst orbits table to store initial, destination, and breakup orbits. It has less constraints than the main orbit tables, since this data is derived at varying degrees of quality depending on the application.

5. OWNER/OPERATOR MODEL

DISCOS adopted a new owner/operator model in 2013[1]. This model continues to be employed, but there was a change in its application due to a limitation with single entities. Throughout DISCOS 2, there were many-to-one relationships to the 'organisations' table. In some cases, composite organisations were created, e.g. "France/Germany". This meant that countries were part of several different entries in the organisations table, hindering the ability to do a reverse lookup for those countries. In DISCOS 3, countries were placed back into their own table, and a new entities base table was created that contains both organisations and countries. Throughout DISCOS 3, entities use a many-to-many relationship via association tables.

CONCLUSION

The DISCOS 3 data model improves on DISCOS 2's solid foundation. The object catalogue's DISCOS ID promotes objects to first-class citizens, and lets the data model reflect how they are used within ESA's Space Debris Office. The temporal mapping and spatial relationships provide a flexible model for objects in space.

Inconsistencies found in DISCOS 2 are no longer possible, providing a robust data source for ESA's space debris activities. Catalogue independence will allow DISCOS to scale in the future, as new data sources can be incorporated with ease, and is already used to seamlessly bring multiple data sources together in a coherent interface.

REFERENCES

1. T. Flohrer, S. Lemmens, B. Bastida Virgili, H. Krag, H. Klinkrad, E. Parrilla, N. Sanchez, J. Oliveira, and F. Pina. DISCOS - Current Status and Future Developments. In *6th European Conference on Space Debris*, volume 723 of *ESA Special Publication*, page 38, August 2013.
2. H. Klinkrad. DISCOS - ESA's database and information system characterising objects in space. *Advances in Space Research*, 11:43–52, 1991. doi: 10.1016/0273-1177(91)90541-Q.