Upgrade of ESA's collision avoidance system at ESOC

Quirin Funke(1), Volker Schaus(1), Frazer McLean(2), Vjeran Wertag(1), Klaus Merz(3), Jan Siminski(3)

(1)IMS Space Consultancy @ ESOC
Darmstadt, Germany
Email: quirin.funke@ext.esa.int

(2)CS Group @ ESOC
Darmstadt, Germany

(3)ESA/ESOC Space Debris Office
Darmstadt, Germany

***Abstract* – Collision avoidance is an integral part of spacecraft operations. It is essential for safeguarding own assets as well as preventing the proliferation of space debris. The European Space Agency's (ESA) Space Debris Office provides operational collision avoidance support to ESA and third-party missions for around 20 years.**

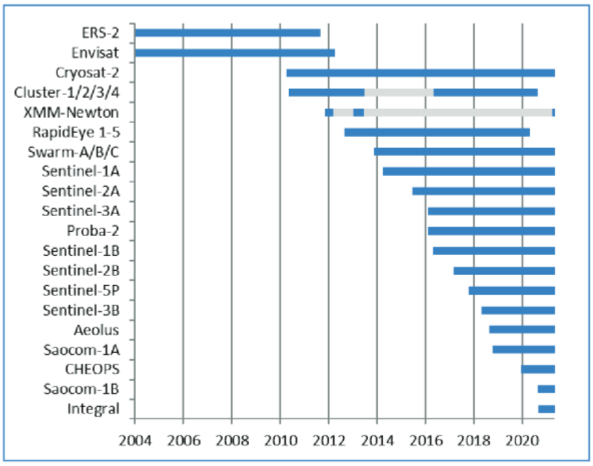
The service has seen continuous evolution, coping with diversifying, and increasing mission needs and platform capabilities, the increase of tracked space debris objects in orbit, but also with the rise of more and better data sources, as well as means to better address uncertainties in the prediction process for conjunction events. Our most recent challenge results from emergence of large constellations in close orbital regions and the acceleration of the number of small satellites in the space traffic.

The overall processing chain is currently modernized to cope with expected future data loads and to be able to ingest conjunction warnings from multiple surveillance data providers. The upgrades aim at making the tool base more modular allowing for a gradual replacement of legacy tools as well as a more flexible process handling allowing for a mix of automated and manual workflows. The chain includes conjunction event detection, collision risk assessment and visualization, orbit and covariance propagation, manoeuvre optimisation, distributed task queue, process control and data handling.

This paper will outline the new developments and present the changes already implemented such as a new process control system and replacement of key flight dynamics components via integration of the new ESA/ESOC Flight Dynamics library GODOT.

I. INTRODUCTION

Collision avoidance is part of routine spacecraft operations in ESA to comply with Space Debris Mitigation guidelines [1]. Operational conjunction analyses and collision avoidance activities at ESA started with ERS-2 and Envisat and nowadays concentrate on ESA's Earth Explorer missions, the Copernicus' Sentinel spacecraft in LEO, ESA's science missions in HEO as well as on external partner spacecraft – see Figure 1 for a history of spacecraft fully supported by the operational process. Additionally, several other missions were covered in the past under varying support level and duration (e.g. LEOP): Proba-1, -V, Galileo/Giove, METOP-A/-B/-C, MSG-3/4, Artemis, Lisa-Pathfinder, Sentinel-6A.



The collision probability assessment is based on conjunction data messages (CDMs) provided by the US 18th Space Defense Squadron (18 SDS). The CDMs contain information on conjunctions between tracked objects in the US catalogue (chasers) and the own operated spacecraft (target) trajectories, in particular time of closest approach (TCA), separations, state vectors and covariances at TCA as auxiliary information on the orbit determination setup and quality. Due to a data sharing agreement between the US Strategic Command (USSTRATCOM) and ESA, signed on October 30th, 2014, the SDO has access to CDMs covering larger volumes around the target trajectories and longer lead times.

The collision risk assessment requires the following processing steps:

- Compute probability of collision using object geometries from DISCOS and the orbital state and uncertainty information from the CDM for the

target and chaser.
- Compute the collision probability using the state and covariance for the chaser from the CDM and the state and covariance from an internal flight dynamics solution.
- Generate a collision avoidance manoeuvre plan in case the probability is above a reaction threshold.
- Search for conjunctions with ephemeris files provided by other operators or an internal catalogue built up from CDM chaser objects.

The processing steps are triggered by new data, i.e. when new CDMs or new flight dynamics ephemeris files are available. Large batches of CDMs were initially provided multiple times per day. Then, the CDMs were processed in a sequential way. To deal with an increased number of CDMs from possibly multiple data providers, a new parallel collision avoidance processing framework needed to be developed. The legacy process is explained first and then details of the upgraded system are provided.

II. Legacy Process

The current processing chain connects the database and various numerical software packages such as CRASS [3] and CORAM [2] as well as auxiliary tools via a collection of interconnected scripts of different types (shell, Perl, Python) triggered by crontabs. It is described in Figure 1 and is described in detail in [4]. This system has become increasingly complex over time e.g. due to many special cases to be covered such as dedicated interfaces for various missions and tweaks to prioritise tasks and increase performance. This complexity has led to significant effort to maintain the system but also significant training needs for the analyst staff operating the system.
It has therefore been decided to refactor this processing chain and related tasks, build a convenient web-based User Interface (UI) for controlling the process, embed procedures in this UI, improve fault tolerance and repeatability and increase performance and scalability.

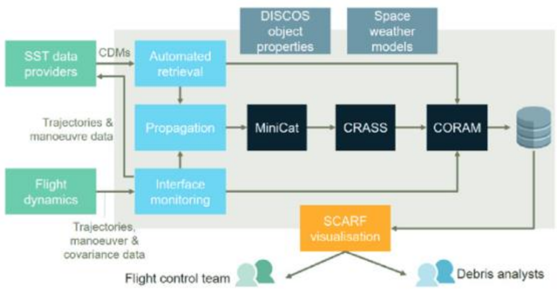


Fig. 1. Legacy Collision Avoidance Process at the ESA/ESOC Space Debris Office

III. Upgraded Process

To achieve this, the new framework is built on distributed computing using Celery [5] task queues with a custom workflow and job implementation. Workflows control the order of jobs, which perform the actual work, and track their progress; they also support interaction with the jobs, like changing priorities. A central scheduler is responsible for starting workflows and keeping track of their status. The system is designed to be scalable as new worker nodes can be easily added and jobs can be worked on in parallel. The system is described in Figure 2.
A web-based UI using Flask [6] in the backend and React [7] in the frontend allows the debris analyst to monitor and control the process chain.

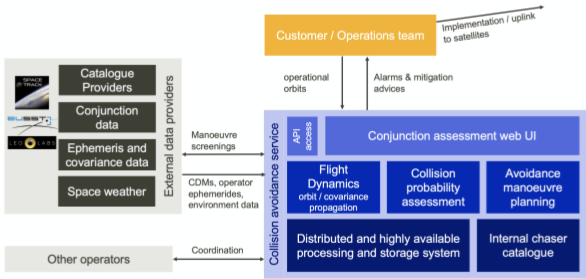


Fig. 2. New Collision Avoidance Process at the ESA/ESOC Space Debris Office

A. *Process control*

By basing the processing system on Celery with a custom workflow and job implementation we achieve scalability and flexibility. Celery's distributed task queue architecture enables us to scale our processing capabilities based on demand, ensuring seamless performance even during peak loads like LEOPs. Physical servers can easily be augmented by cloud resources.
As Celery’s built-in workflow system (canvas) is relatively stiff, we went for a custom workflow and job implementation. This gives us the freedom to tailor the system to our specific needs, allowing for efficient task orchestration and streamlined operations.
Workflows can either be started manually via the user interface (UI) or command-line interface (CLI), with automated time-based schedules, or when new orbit files are delivered. This offers a robust and adaptable solution for managing workflows across a diverse range of scenarios and requirements, including LEOPs and other special operations.

B. *Data handling*

The data handling revolves around a robust infrastructure centred on a PostgreSQL database and MinIO for S3-like file storage. Through data mirroring, this setup ensures efficient and reliable storage of both structured and unstructured data generated throughout the system's operation. A notable feature is the comprehensive persistence of all job inputs and outputs, stored within the central database and MinIO, respectively. This approach enables seamless reprocessing and debugging by providing a complete historical record of data transformations and outcomes.

C. Tools

A significant part of the development of the new framework and its workflows and jobs consists in encapsulating clear interfaces to the functionalities provided by the core numerical tools. Therefore, this activity also paves the way for the opportunity to replace legacy code one by one. This has already started with the replacement of CORAM with CORAL, a new collision avoidance library, based on the Godot astrodynamics platform currently being developed by ESOC FD with contributions from SDO.

D. Interface

The web-based UI is built with React components and designed mobile-friendly with Tailwind CSS. A core concept is to support all configuration and tasks via the UI, also from mobile devices. The progress and logs of workflows and their jobs can be monitored live. Figure 3 shows a screenshot of it.

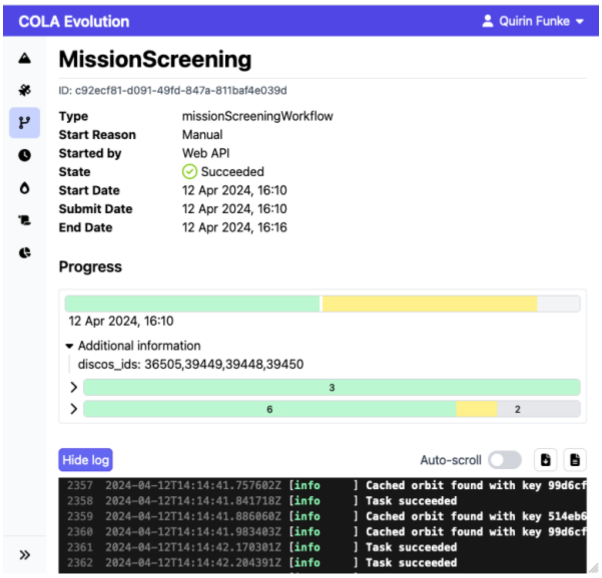


Fig. 3. Workflow display UI

IV. Summary and Conclusions

ESA's Space Debris Office provides operational collision avoidance support to internal ESA missions as well as external partners. In this paper the current operational process has been described with a focus on how the building blocks connect to an overall largely automated process flow. As these connections have been implemented over time resulting in a complex system of scripts, the processing framework is currently being refactored. The modern software technologies used have been outlined, allowing to improve fault tolerance and repeatability, to increase performance and scalability and to provide a convenient UI for controlling the process.

V. References

[1] ESA Space Debris Mitigation Requirements (ESSB-ST-U-007 Issue 1)

[2] Grande-Olalla, I., Sanchez-Ortiz, N., Pulido, J. A., & Merz, K. (2013, August). Collision Risk Assessment and Avoidance Manoeuvres for ESA. 6th European Conference on Space Debris, Darmstadt, Germany.

[3] Klinkrad, H., Alarcon, J. R., & Sanchez, N. (2005). Collision avoidance for operational ESA satellites. 4th European Conference on Space Debris, Darmstadt, Germany

[4] Merz, K., Virgili, B. B., Braun, V., Flohrer, T., Funke, Q., Krag, H., Siminski, J. (2017). Current collision avoidance service by ESA's space debris office. 7th European Conference on Space Debris, Darmstadt, Germany.

[5] Celery Project (2021, April). Celery – Distributed Task Queue. https://docs.celeryproject.org/

[6] Pallets Project (2021, April). Flask documentation. https://flask.palletsprojects.com/

[7] Facebook Inc (2021, April). React - A JavaScript library for building user interfaces. https://reactjs.org/