Enabling NewSpace Innovation Thanks to Off-the-shelf Flight Dynamics Systems

Benoît P. REMY[(1)], Vincent AZZOPARDI[(1)], Thomas PHILIPPE[(1)] , Jesús ESTEBAN DONES[(1)] , Stéphanie HOUDELIN[(1)] , Thierry WARROT[(1)] , Stéphanie MAREL[(1)] , Clément BÉAL[(1)] , Michel LACOTTE[(1)]

[(1)]Centre National d'Études Spatiales (CNES)
Toulouse, France
Email: benoit.remy@cnes.fr

***Abstract* – Space Flight Dynamics Systems (FDS) development and operations are inherently and inevitably complex. They require not only deep expertise in diverse fields (space flight dynamics itself, applied mathematics, computer science, front- and back-end software development, configuration management, IVV, etc.), but most importantly their overall orchestration.**

As such, FDS development may be a burden and a threat to the success of NewSpace businesses who ambition to become spacecraft operators.

Based on more than 40 years' legacy and best practices in FDS development and operations, CNES has developed and offers the fully generic and scalable SIRIUS FDS product line, together with the associated software factory. Thanks to their extreme interoperability, they may be executed on virtually any environment, architecture and OS, from laptops to cloud infrastructures; for highly critical applications, deep space missions, or large constellations; now or 25 years from now.

While customized versions are in operations for many years on highly demanding missions, the off-the-shelf standard FDS is, for the first time, used operationally in 2023 for a small satellite. Within the coming months, the number of in-orbit satellites operated with either generic or mission-specific SIRIUS FDS will exceed 30.

SIRIUS thus enables disruptive space businesses to focus on what they do best: innovation.

I. INTRODUCTION

The structure of the present article is as follows:

- Firstly the technical and programmatic enablers of the FDS products and the FDS product line.
- Secondly the achievements and the feedback from three ongoing or forthcoming NewSpace missions.
- Thirdly, perspectives on the product line roadmap and some new functionalities to be expected.

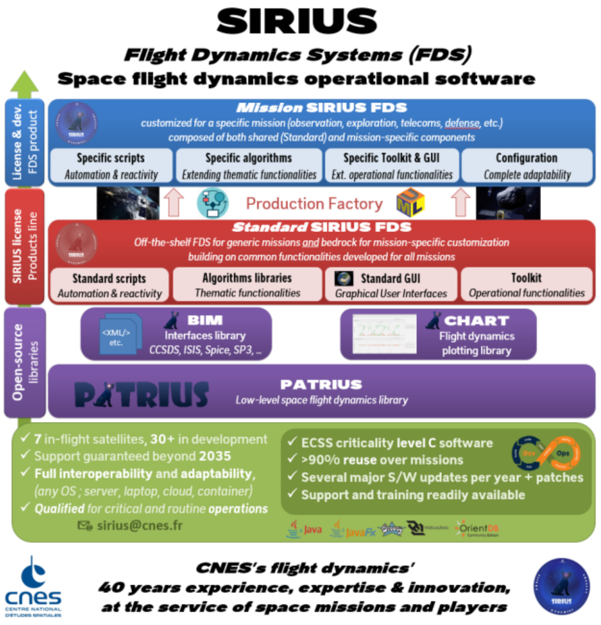
II. TECHNICAL AND PROGRAMMATIC ENABLERS

A. *Functionalities and requirements*

The SIRIUS FDS products have an extensive list of **functionalities**, derived from both mission-specific requirements, and generic requirements, the latter concentrating the union of the former over the years and over the different missions using SIRIUS and its predecessors [1].

As presented previously, the generic "standard" FDS is an off-the-shelf product that fulfils the generic requirements. These generic **requirements** are such that they typically fulfil 90 to 100% of a given mission's requirements, even the most demanding.

NESS was the first mission to employ the standard FDS as-is, proving the **off-the-shelf** concept for simple missions, which is of particular interest for NewSpace applications. For complex missions however, the typical process is –and will continue to be– to develop a **custom** product, largely based on the standard FDS.


SIRIUS
Flight Dynamics Systems (FDS)
Space flight dynamics operational software
License & dev. FDS product
Mission SIRIUS FDS
customized for a specific mission (observation, exploration, telecoms, defense, etc.)
composed of both shared (Standard) and mission-specific components
Specific scripts
Automation & reactivity
Specific algorithms
Extending thematic functionalities
Specific Toolkit & GUI
Ext. operational functionalities
Configuration
Complete adaptability
Production Factory
SIRIUS license Products line
Standard SIRIUS FDS
Off-the-shelf FDS for generic missions and bedrock for mission-specific customization
building on common functionalities developed for all missions
Standard scripts
Automation & reactivity
Algorithms libraries
Thematic functionalities
Standard GUI
Graphical User Interfaces
Toolkit
Operational functionalities
Open-source libraries
<XML/> etc.
BIM
Interfaces library
CCSDS, ISIS, Spice, SP3, ...
CHART
Flight dynamics plotting library
PATRIUS
Low-level space flight dynamics library
✓ 7 in-flight satellites, 30+ in development
✓ Support guaranteed beyond 2035
✓ Full interoperability and adaptability, (any OS ; server, laptop, cloud, container)
✓ Qualified for critical and routine operations
sirius@cnes.fr
✓ ECSS criticality level C software
✓ >90% reuse over missions
✓ Several major S/W updates per year + patches
✓ Support and training readily available
Dev
Ops
Java
JavaFx
OrientDB
cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES
CNES's flight dynamics'
40 years experience, expertise & innovation,
at the service of space missions and players
Figure 1: SIRIUS development layers.

The **Java** programming language, development kit (JDK) and virtual machine (JVM) powers all levels of the FDS, providing firstly a high degree of abstraction for use and development, and secondly a complete interoperability.

UML or UML-like **models** consist a large basis of the development, in particular for the definition of datatypes, services, and GUI widgets. Auto-generated code and available libraries (including CNES's open-source PATRIUS [4]) reduce significantly the development and customisation effort [3].

A **Python scripting API** allows direct scripting and orchestration in Python and Bash/Shell CLIs, through Websockets (Websockets API also accessible directly).

Any SIRIUS FDS version is releasable as standalone, RPMs, and/or containers package, allowing easy yet robust configuration management, repeatability, and reliability. It runs on virtually any **environment**, from laptops to cloud servers. It is also a fully integrated component of (yet completely independent from) the CNES's ISIS control centre product line.

Users have access to an extensive **documentation**:

- Developers: Javadoc and API documentation, directly in the provided custom Eclipse-based Integrated Development (IDE) and Modelling Environments, and;
- Operators: standalone wikis, thematic documentation, and data documentation, interactively in the GUI itself.

Training and training material is readily available (15 different sessions with video training material).

Multiple layers of **validation** induce a very high reliability of the products: unit, integration, system, functional, and operational tests with level-C criticality requirements as defined by CNES and ECSS standard.

Finally, the use of the **best practices and tools** enable continuous integration/development (CI/CD) processes, ensuring efficient project management, development, deployment, integration, and validation.

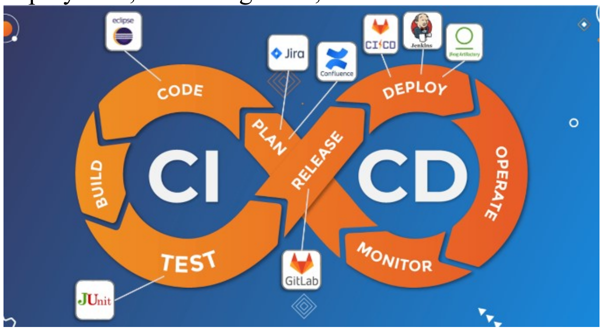


Figure 2: SIRIUS CI/CD pipeline and main supporting tools.

B. Architecture and components

SIRIUS FDSs contain:

1. interfaces (input/output files, streams, or GUI)
    a. CCSDS, ISIS, XML standards, etc.
2. data (a database):
    a. object-oriented data
    b. references between data
    c. tree structure
    d. persistence (history & exports)
3. processes:
    a. interfaces operations (ingestion or generation)
    b. data processing algorithms
    c. database operations

Data (objects with inheritance) is organized in a tree structure, in a *context* that has *branches*. Objects may reference each other.
Similarly, processes are also organized in a tree structure where they can be configured and executed. Every execution produces a process record that can be re-run.

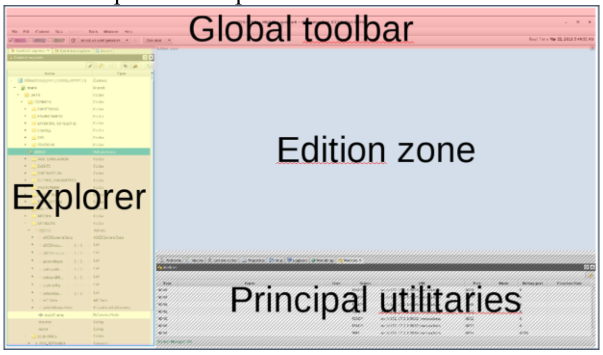


Figure 3: Default GUI view.

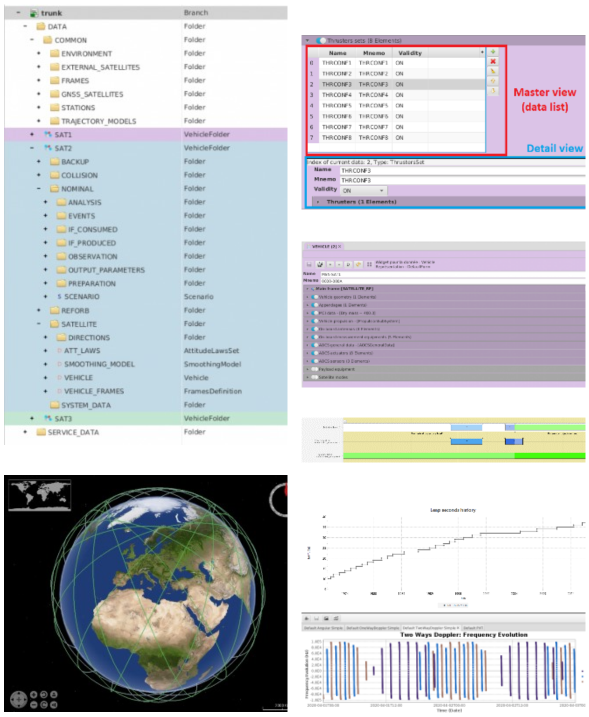


Figure 4: Random GUI views: Context explorer (top left); default data editor views (top right), a scenario editor view (centre right), 3D view (bottom left), graphs (bottom right).

Stages run in sequence or in parallel to achieve the required **processing** or generate the required interfaces. Each stage is guaranteed to be coherent; the status of the system is known; repeatable; and restorable.
These typically include:

1. Data ingestion;
2. Determination (orbit, attitude, and generally the S/C state) of the past;

3. Prediction (idem) of the future;
4. Planning (and its translation into actuation) to modify the future; closed loop with prediction;
5. Data generation.

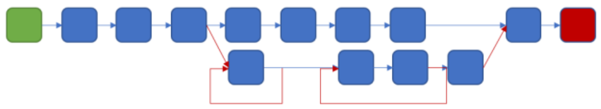


Figure 5: Example of FDS processing stages sequence

Several independent components work together through their own API. These are:

- the orchestrator (scripting API), for automatic operations;
- the graphical user interface(s) (GUI) for manual or semi-manual operations;
- the workers manager, to accommodate for several users and parallel stages;
- the workers themselves, that run from coherent (and reproducible) state to the next one;

Different machines may host different components. The two most common installations are:

- In control centres, there is typically a central server and one client (GUI) per operator.
- In standalone installations (e.g. laptop install), all components run on the same machine.

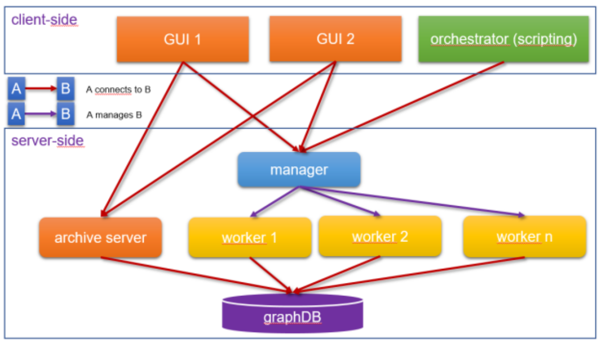


Figure 6: SIRIUS FDS components.

Additional external tools can also be plugged in, for instance for data expertise or visualization (such as CNES's VTS visualization tool for space data).

C. Configuration

Each component is fully configurable. Configuration generally consists of key-value pairs stored in XML files, which include (non-exhaustive list):

- environment & interface: IP, ports, files, formats, standards, memory, timeouts;
- logging: level, processes, satellites, multi-files, rollback, naming;
- GUI: colouring, satellites, pagination, profiles;
- locale: English and French are implemented product-wide, custom time formats, units, etc.

All configuration related to the algorithms themselves is stored in the database, to ensure its consistency.

D. Scripts

Scripts, which are executed via the scripting API, are written in Groovy (which is an extension of the Java language), and may take advantage of the whole libraries and functionalities of the product: retrieve data, execute a custom processing, execute a processing stage, display results, generate files or synthesis, etc.

E. Graphs and Synthesis

Graphs are indifferently generated in the GUI or by the server; and can be saved into various file formats.
Custom HTML synthesis may be created and tailored to the mission needs. Available libraries, templates, and default implementations include orbit determination, prediction, manoeuvres, and planning syntheses.

F. Development Processes and How to Integrate Them

Any new FDS development will typically start by using the latest available standard FDS (and the underlying SIRIUS libraries) and identifying missing needs. The mission may them decide to fork into a branch corresponding to that standard FDS at any point in time, and develop its own product.
It is also possible for any mission to upgrade itself to any earlier version of the standard, to benefit from improvements brought since. This process is usual, and the SIRIUS product line provides the corresponding tooling, in particular the migrators to migrate a database from one version to another, should need be.

In the meantime, all missions interested in the progress of the standard are:

- offered visibility on the SIRIUS issue tracking system (problem reports, change requests, activities) and collaborative edition system (good practises, documentation, etc.);
- invited to participate to the definition of its releases, which will namely include cherry-picking evolutions from the other branches.

While each mission benefits to the others, it also benefits from the others.

III. Achievements and Feedback

On top of proving to be a key element in the success of numerous critical missions, SIRIUS FDSs also proves its adequacy for NewSpace missions, as demonstrated by the following examples.

A. *NÉSS*, a 3U interference detection demonstrator CubeSat, launched on 9 October 2023.

- NεSS was the 1st satellite operated directly using the off-the-shelf standard FDS;
    - The only customisation consisted in configuration and scripting.
    - Already fully validated by other missions, only system validation was required.
    - Benefited 'bonus' functionalities, not anticipated nor strictly necessary, yet very useful during operations, e.g.:
        - History, graphs & synthesis generation; on-the-fly interface generation, etc.
        - Accurate orbit determination capability even during safe mode or with GNSS off.
- NεSS's FDS became fully automated immediately after LEOP, less than 36 hours after separation. No human intervention is required since.

These aspects implied large cost, time, and risk reductions during all phases of the mission (development, validation, and operation).

B. ***KINEIS**, a 25-satellites Internet-of-Things global coverage commercial constellation.*

Planned for launch, in five batches of five over a period of 10 months, the KINEIS mission shall require a high degree of automation and operability.

Indeed, it will undergo multiple parallel operations on an increasing number of satellites in different concurrent phases: launch & early operations (LEOP), commissioning, routine, and decommissioning.

To ensure both global coverage and availability necessitating a significant amount of electric propulsion (EP) periods for orbit maintenance, the KINEIS FDS is key to the success of the mission.

Its ability to produce web syntheses autonomously for the whole constellation provides easy monitoring for the operators, allowing them to monitor, visualize, analyse, and respond quickly to any situation.

C. *YODA, a 2-satellite geostationary patrol demonstrator for the French Space Command*

Contrarily to its predecessors [2] that typically followed a V-model development lifecycle at mission level, YODA's FDS –as well as the overall YODA mission– is currently under development in an agile development approach, allowing quick iterations between developers & operators, and efficient development with evolving requirements and constraints.

IV. Perspectives

On top of the ongoing missions, more and more missions, both public and private, of all types, are operated by SIRIUS FDSs. The coming years will also see the first use of SIRIUS in interplanetary operations, and hopefully the implementation of promising toolkit features in the fields of UX design and thin clients, performance, and cybersecurity, not to mention always-expanding space flight dynamics libraries.

V. Conclusion

As a conclusion, a number of technological, technical, and programmatic layers allow SIRIUS, CNES's flight dynamics systems product line, to offer a mature (TRL9) off-the-shelf standard FDS, perfectly suited for NewSpace applications requiring fast time-to-market, agility, interoperability, and reusability.

VI. References

[1] R. Houdroge, D. Claude, J. Anton, T. Sabatini, P. Cardoso, G. Mercadier, T. Trapier, Y. Tanguy, 'The SIRIUS Flight Dynamics Library for the Next 25 Years", *5th ICATT*, The Netherlands, 2012.

[2] GMV, Y. Tanguy, M. Lacotte, JJ. Wasbauer, "SIRIUS-DV: The new Flight Dynamics algorithms for the future CNES missions", *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*, Germany, March 2016.

[3] P. Annat, R. Bernard, J. Esteban Dones, "SIRIUS Model-Driven Software Product Line for Flight Dynamics Systems", *31st ISTS, 26th ISSFD, 8th NSAT*, Japan, June 2017.

[4] JF. Goester, "Free Java CNES Flight Dynamics Tools", *7th ICATT*, Oberpfaffenhofen, Germany, Nov. 2018.