Community expertise in Science Ground Segments: Planck, Euclid and beyond

A. Zacchei, T. Gasparetto, D. Tavagnacco (INAF - OATs)

What is “SGS” – in ESA missions

- The SGS is charged with many tasks…

Survey Planning	Data Processing
Instrument operations	Data archiving and Science Support
Quick Look Analysis	Simulations
Calibrations	PA/QA

- …and made by different actors
  - Science Operation Centre
  - Science Data Centre(s)
  - SW developers / DA experts
  - Instrument Operation Team

- …and receiving inputs from
  - MOC
  - Possible feedback form User Community

What is “SGS”

YES,

- Machines
- Storage

BUT more important

- **People**
- SW framework
  - versioning, libraries, archive
- Simulation + Analysis SW
  - versioning/storage
- Instrument knowledge
- Pipeline(s) [i.e. groups of SW]
  - validation/versioning
- Management (coordination)
  - Science
  - Infrastructure (tech)
  - Instrument(s) “IOT”
  - working groups
  - simulations

Planck

- Two Planck DPC (Data Processing centers) have been responsible of the operations and data analysis. Both follow the same overall approach to the data reduction.

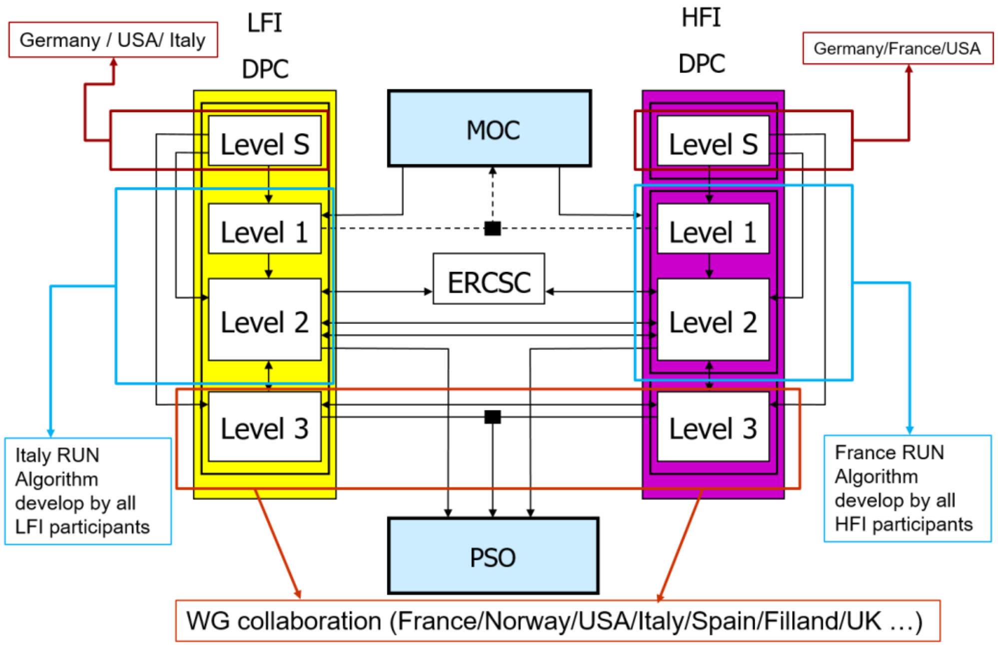
- In the initial design phase for efficiency/redundancy/cross-checking purpose was proposed that each DPCs should have been able to analyze the data of the other instruments. Software layer was built to cover this requirement but it reveal to be too complex. Only Level S was keep common. Fixed interfaces has then been defined to exchange data at each level.

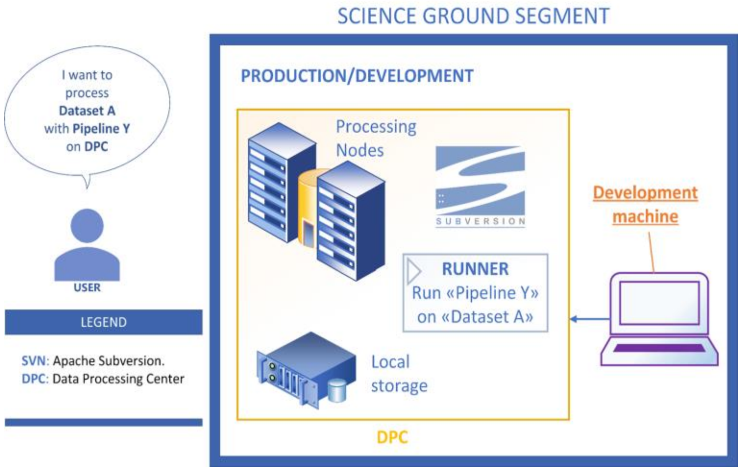
- Process has been then logically divided in four main levels:
  - Level 1 responsible to get directly the data form MOC, produce the DQR, operate the Instrument, transform HK and Science telemetry in raw timeline and store in a dedicated database;
  - Level 2 was dedicated to synthesize the instrument information in the IMo, remove the systematics, flag not usable data, calibrate and finally create the maps and all associated products;
  - Level 3 was dedicated to separate components into catalogues and specific astrophysical emissions.
  - Level S responsible to produce the required simulation needed to validate the pipelines.

- For every essential step of a pipeline for which a proven method does not exist, we develop at least two independent methods. We promote cooperation over competition. Each step was internally validated and most of the DPCs time was spent to cross check all the results first internally and then between instruments.

- The decision to have two different pipeline developed independently at each DPC add **strong** value to the cross-instrument validation.

Planck - Implementation





Euclid

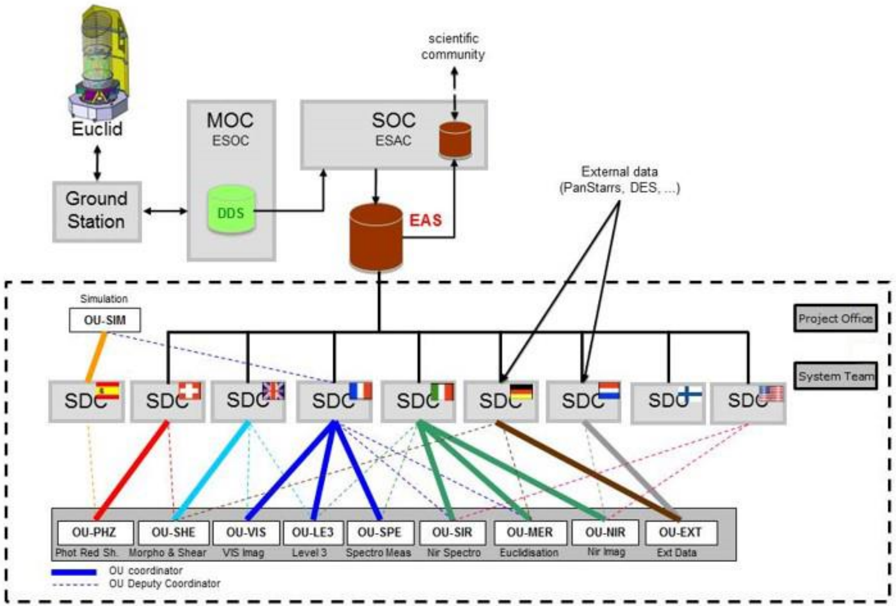
- Euclid will produce and use a big amount of data (estimated to be at the end of the mission of the order of hundred PB). It will be then essential to avoid excessive data transfer, to develop a structure where the code will be moved instead of the data.
- The various Science Data Center are providing different hardware

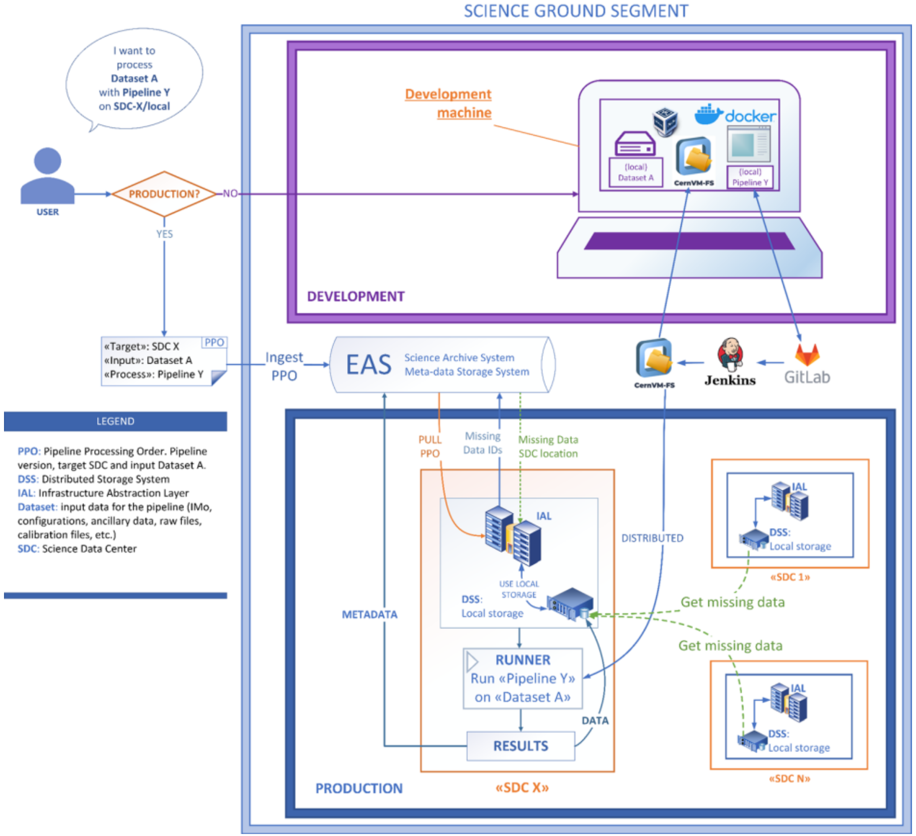
then

- Two languages (C++ and python) has been selected a Common Data Model and Common Sw Infra has been built → same code should be executed in any Science Data Center to allow parallel process and redundancy.
- The creation of a common infrastructure is very manpower demanding and require lot of test. For this reason in Euclid we set different Instrument technology test that verify the entire infrastructure in each SDC. At the same time to facilitate the scientific code integration we institute the year based Developers Workshop with the aim to be a tutorial for Euclidian developers.

- The Data processing pipeline in Euclid are a series of Processing Functions: designed by the OUs (Organization Units, scientist), developed in collaboration between the OUs and SDC developers , integrated by the SDCs, and running on the SDCs infrastructure.
- Processing Function has been tested in to the SGS in growing complexity.

Euclid - Implementation





SGS Evolution - Lessons Learned

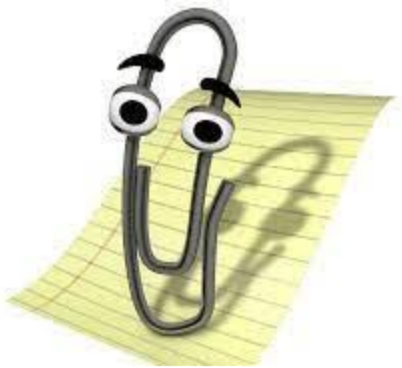
Space Missions = Large Datasets + Large collaborations

Space missions will produce larger datasets to be analyzed (mainly simulations) by a large number of teams that need to be **coordinated**. The SGS (efficiency) **is a crucial part of the mission** dealing with**:**

- **common processing environment, infrastructure scalability** with easy **maintenance**
- robust interconnections to create, distribute, version and use the software
- optimized **data transfer** and sharing

In the evolution of the SGS, it's mandatory to take into account:

- multiple **Processing Centers**
- **storage** may not be centralized, but distributed among Processing Centers
- **code** available and executable in all Processing Centers ensures **consistency**
- limited **data transfer**
- allow for many **programming languages** (including newer ones)
- interchangeable development and production **environments**
- simple **infrastructure** management



SGS Design

Ingredients to design an SGS:

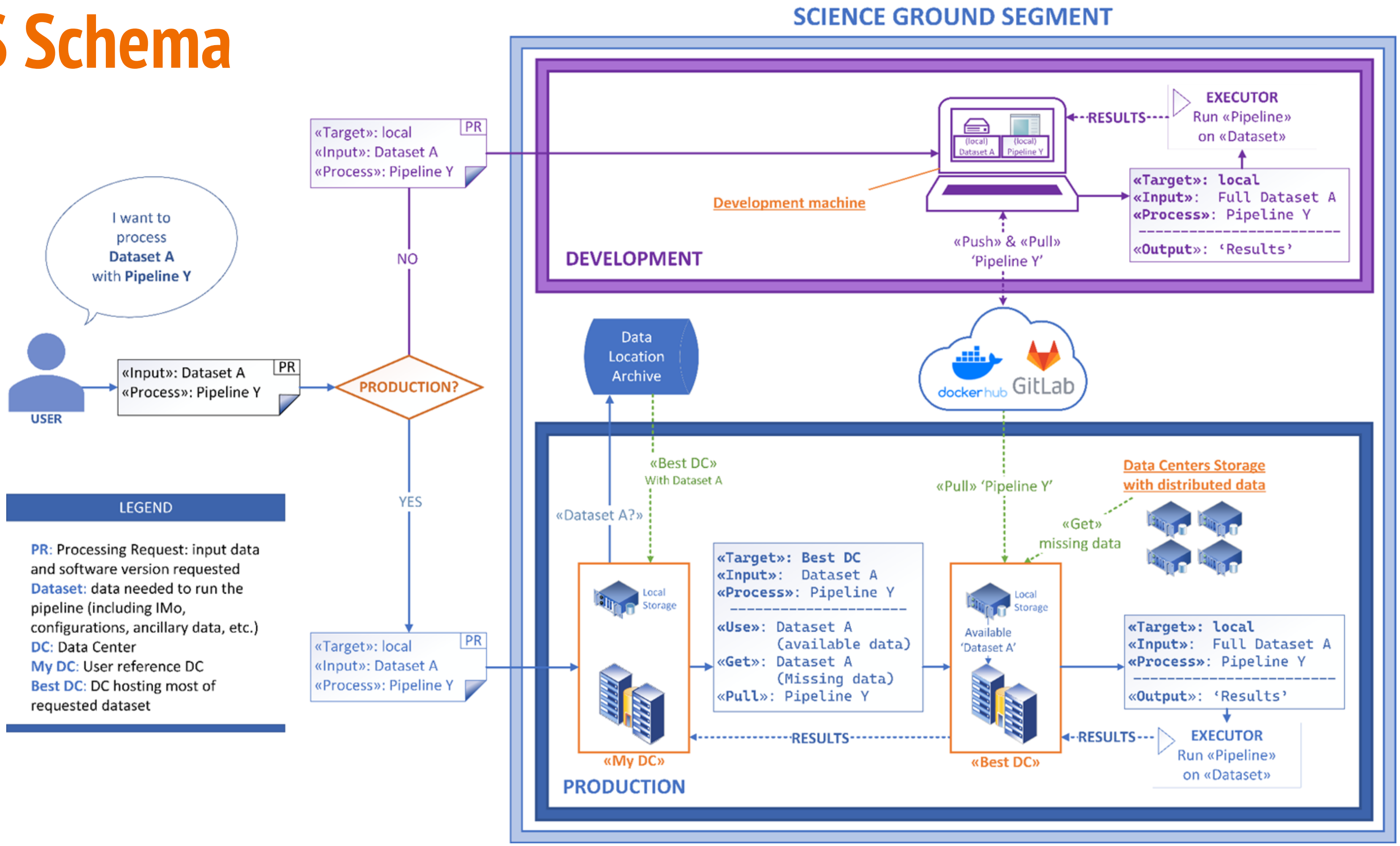
- Identify **interfaces**:
  - dataset (volume, complexity)
  - processing steps (SW modules)
  - understand how to group/divide data in modules
- Define the **environment**:
  - collaborative tools
  - versioning tools and information transfer (reproducibility)
  - maximize flexibility
- **Abstract** the **SW** infrastructure from **HW** infrastructure
- Design the infrastructure aiming for maintenance and upgrade over the time span of the mission

SGS “today” implementation

Using nowadays tools, like **Docker** and **GitLab**, is possible to:

- **versioning** for a long period of time
- **abstract** SW and HW infrastructure
- support **many programming languages**
- define a **common environment** for development and production
- **distribute** the SW allowing each “module” to be a usable “black box”
- guarantee **reproducibility** and consistency
- easy **maintenance** and upgrade of infrastructure

SGS Schema

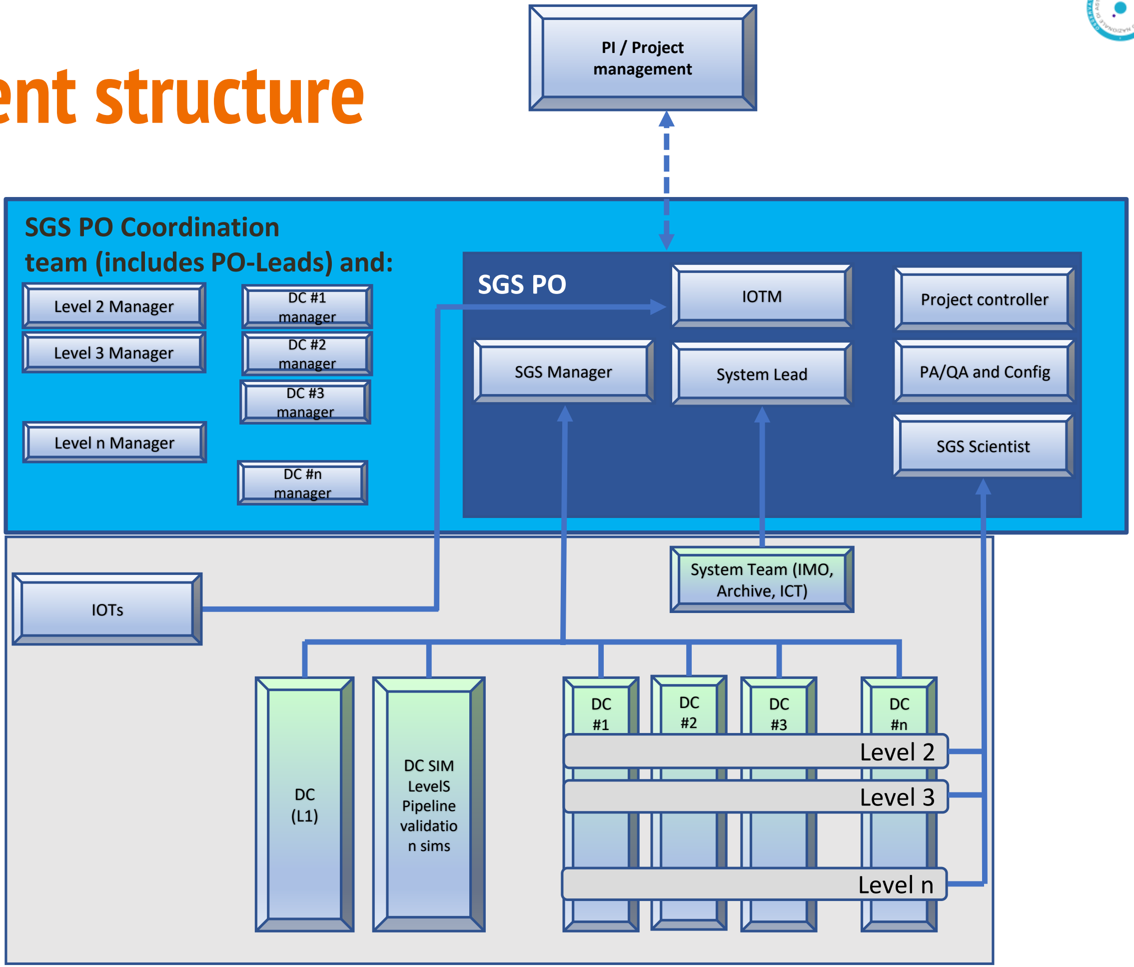


SCIENCE GROUND SEGMENT
DEVELOPMENT
(local) Dataset A
(local) Pipeline Y
Development machine
RESULTS
EXECUTOR
Run «Pipeline» on «Dataset»
«Target»: local
«Input»: Full Dataset A
«Process»: Pipeline Y
«Output»: 'Results'
«Push» & «Pull» 'Pipeline Y'
«Target»: local
«Input»: Dataset A
«Process»: Pipeline Y
PR
I want to process Dataset A with Pipeline Y
USER
«Input»: Dataset A
«Process»: Pipeline Y
PR
NO
PRODUCTION?
YES
Data Location Archive
docker hub
GitLab
«Best DC» With Dataset A
«Dataset A?»
«Pull» 'Pipeline Y'
Data Centers Storage with distributed data
«Get» missing data
Local Storage
«Target»: Best DC
«Input»: Dataset A
«Process»: Pipeline Y
«Use»: Dataset A (available data)
«Get»: Dataset A (Missing data)
«Pull»: Pipeline Y
Local Storage
Available 'Dataset A'
«Target»: local
«Input»: Full Dataset A
«Process»: Pipeline Y
«Output»: 'Results'
EXECUTOR
Run «Pipeline» on «Dataset»
RESULTS
RESULTS
«My DC»
«Best DC»
PRODUCTION
«Target»: local
«Input»: Dataset A
«Process»: Pipeline Y
PR
LEGEND
PR: Processing Request: input data and software version requested
Dataset: data needed to run the pipeline (including IMo, configurations, ancillary data, etc.)
DC: Data Center
My DC: User reference DC
Best DC: DC hosting most of requested dataset

SGS general Mangement structure

PO is the team where different manager discuss all the SGS aspects (Scientfic req, Instrument effect, Infrastructure, PA/QA, schedule) and is responsible of entire organization.
PO reports directly to the Project Mamagenet / Mission management.

- DC are responsible of integrate/execute in a common environment various analysis pipeline.
- Special DC is dedicated to L1 normally near to the Mission operation center.
- Special DC is dedicated to Simulation
- IOT is team in charge to follow instrument development and operations.
- Levels are responsible to define / prototype pipeline aimed at satisfying the scientific req. those are integrated cross DC.



Conclusion - "Why in SGS?"

PRO:

Resources guaranteed

Tech support (common)

Code exchange (calls are standard)

Possibility to use any code

Access real time data

Free developing/prototyping but duty to main goal

long term maintenance

coding rules + PAQA

code documentation (requirements, user manual,..)

CONS:

Resources to be found outside

No tech support

No simple code exchange (must learn every code calls/configuration)

Focus on personal analysis step

Access only consolidated data approved by consortium

free code developing/prototyping

maintenance up to the developer

no coding rules and PAQA

code documentation up to the developer

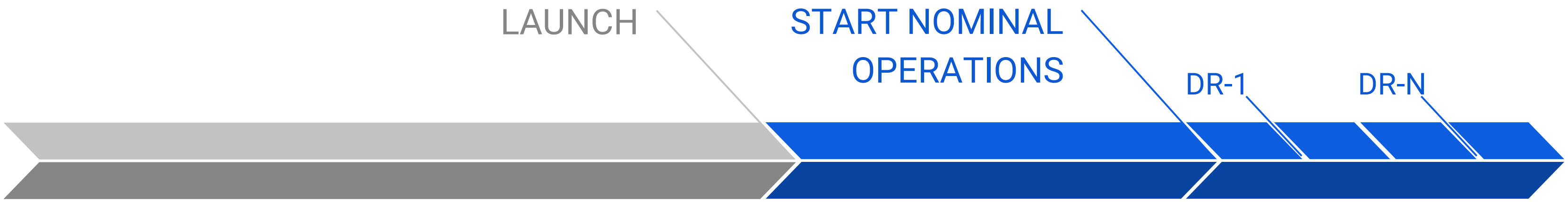
Conclusion - CMB Space Mission SGS

What highlighted before is an example of SGS valid for any mission.
A CMB space mission SGS can be simplified accounting for:

L1 limited data volume → **data** can be distributed in DC
Simulations large volume → **code** can be distributed

- L1 data: they are not that heavy (10-50 TB) and can be distributed to everyone inside the SGS.
- Simulations: distribute SW and configurations to reproduce them.
- Code: different language to be allowed to stimulate collaboration and new ideas.
- Data Model (DM) and Instrument Model (IMo): should define the data structure and the Instrument characteristics to be passed/used by pipelines.

Mission Life Cycle from SGS point of view



	Pre-launch phase (A/B/C/D)	Commissioning / Performance and Verification (E)	NOMINAL OPERATIONS (E, F)
CODE CHANGE	Regular? (science-driven)	FAST! (data-driven)	Slow (analysis-driven)
VERSIONING	Strong (data)	Weak	Strong (code)
INFRASTRUCTURE STRESS	Volume	Flexibility	Load + reliability