Lessons learned from
Gaia data processing & analysis

Jordi Portell (IEEC/ICCUB)

LISA Spain Meeting

ICE-CSIC-IEEC, 15-October-2024

Finançat per

Unió Europea
Fons Europeu
Next Generation

GOBIERNO
DE ESPAÑA

Plan de Recuperación,
Transformación
y Resiliencia

Next Generation
Catalunya

Generalitat
de Catalunya

IEEC INSTITUT D'ESTUDIS ESPACIALS DE CATALUNYA

UNIVERSITAT DE BARCELONA

ICCUB Institut de Ciències del Cosmos UNIVERSITAT DE BARCELONA

EXCELENCIA MARÍA DE MAEZTU 2020-2023

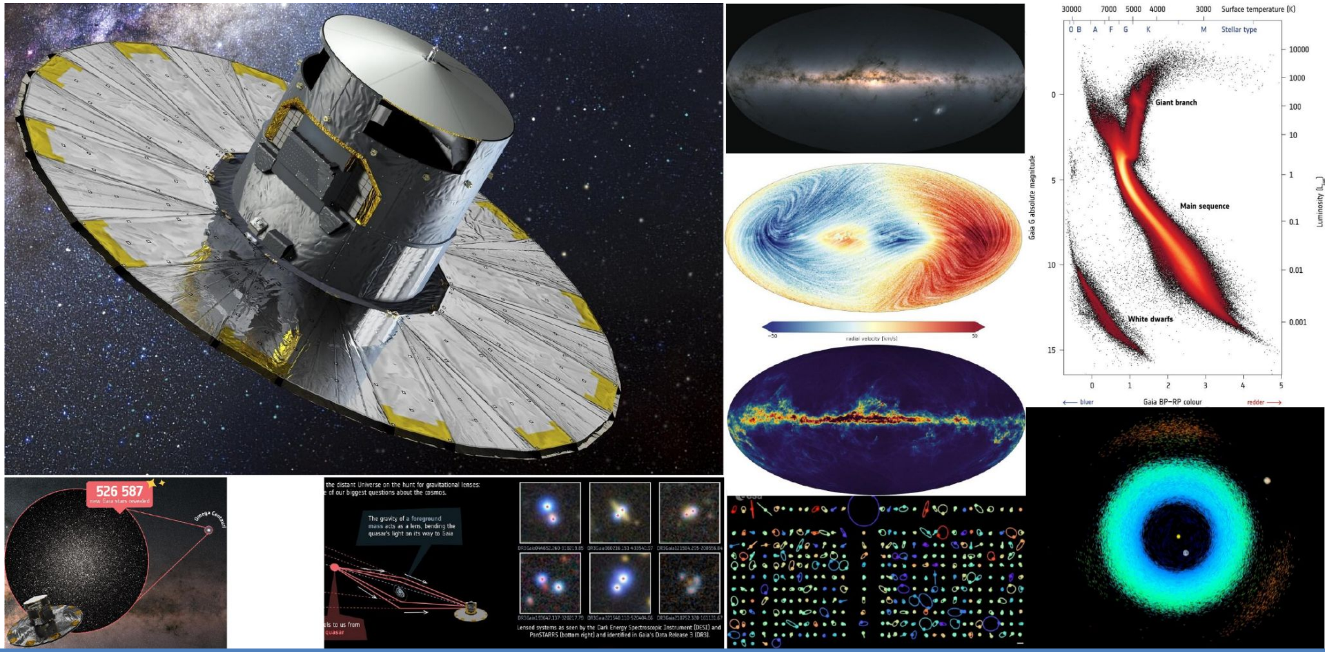
ESA | NASA lisa

gaia

Gaia DPAC Data Processing & Analysis Consortium

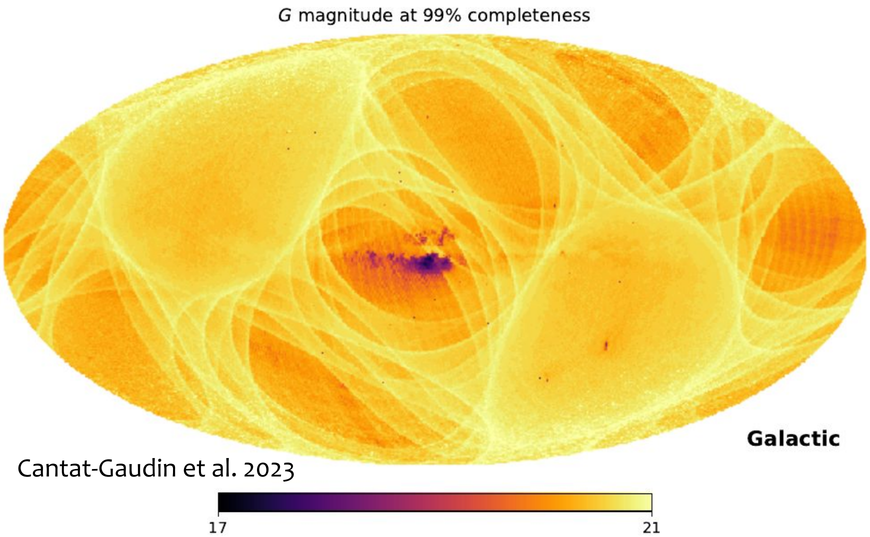
Overview of Gaia

- Global Space Astrometry, chemo-dynamical history of the Milky Way
  - Satellite orbiting around L2 (1.5M km)
  - Spinning+precession: full sky scan, 2 telescopes in the visible spectrum
  - Astrometry + spectro-photometry for >1 billion sources
  - Spectroscopy for a significant fraction of these (>100 million)
- **Discovery machine**
  - Autonomous detection onboard of any point-like source brighter than ~20mag



Selection function

- **Discovery machine**
  - Observing any source within the *Gaia Selection Function*:
    Scan law, brightness, source extension, colour, multiplicity, onboard events, downlink or ground segment issues, …



Cantat-Gaudin et al. 2023

- **LISA will also be a discovery machine!**
  - *Listening* to any source within the *LISA Selection Function*
    →good to start defining it: envisaged limitations and capabilities: frequency band, distance to source, duration of event, sampling, kind (waveform) of event, noise features, orientation/localization, duty cycle, …
  - Prepare for possible failures, malfunctions or under-performance of some onboard systems

Tons of data

- Quite large **data volume**, yes:
  - 136 TB raw data from the spacecraft
  - About 1 PB of reduced data in the Main DataBase (MDB)
- But in Gaia, the main challenge is the **number of records**
  - 258 billion transits (star measurements)
  - **2.5 trillion** astrometric measurements
  - 50 billion high-resolution spectra
  - About 2 billion sources
  - (We don't really deal with *images* in the "classical" sense)

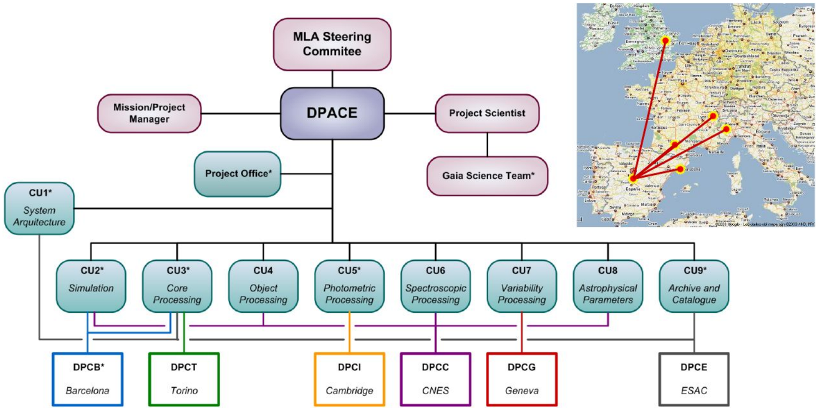
CURRENT DATE AND TIME	2024-10-14T20:39:05 (TCB)
MISSION STATUS	
Satellite distance from Earth (in km)	1,595,107
Number of days having passed since 25 July 2014	3734
Number of days in mission extension	1917
OPERATIONS DATA (collected since 2014/07/25)	
Volume of science data collected (in GB)	136,287
Number of object transits through the focal plane	258,876,153,148
Number of astrometric CCD measurements	2,551,779,223,881
Number of photometric CCD measurements	512,748,606,080
Number of spectroscopic CCD measurements	50,123,773,560
Number of object transits through the RVS instrument	16,840,769,964

- Furthermore:
  - Strongly interrelated data
  - Complex calibration models

  →Iterative (and cyclic) data reduction approach
- Maybe not really comparable to LISA:
  - Not "autonomous observations/measurements" onboard
  - LISA will be more "pure number-crunching" than Gaia → estimate resources (Gaia computing is quite dominated by input/output)

Complex data reduction challenge

- Data Processing and Analysis Consortium, organized in
  - Coordination+Development Units (CUs/DUs) →**algorithms & software**
  - Data Processing Centers (DPCs) →**hardware & operators**
- Each CU, DU and DPC with a (quite) clear goal
  - Not much overlap between them,
    but strong inter-dependencies
    →strict schedule required
    (avoid propagation of delays downstream)
  - Sometimes "politics" may affect in the CU/DU/DPC definition
    →try to harmonize politics with science+technology capabilities/expertise



- **Embrace the Bubble**
  - Some core systems may *need* to "live in a bubble":
    Nearly-ideal conditions, data, models, etc.
    (the conceptual/scientific/technical challenge may be extremely demanding)
  - Some/many systems basically exist to "create this bubble"
    (minimize instrumental effects, issues in data, …)

Technical considerations (I)

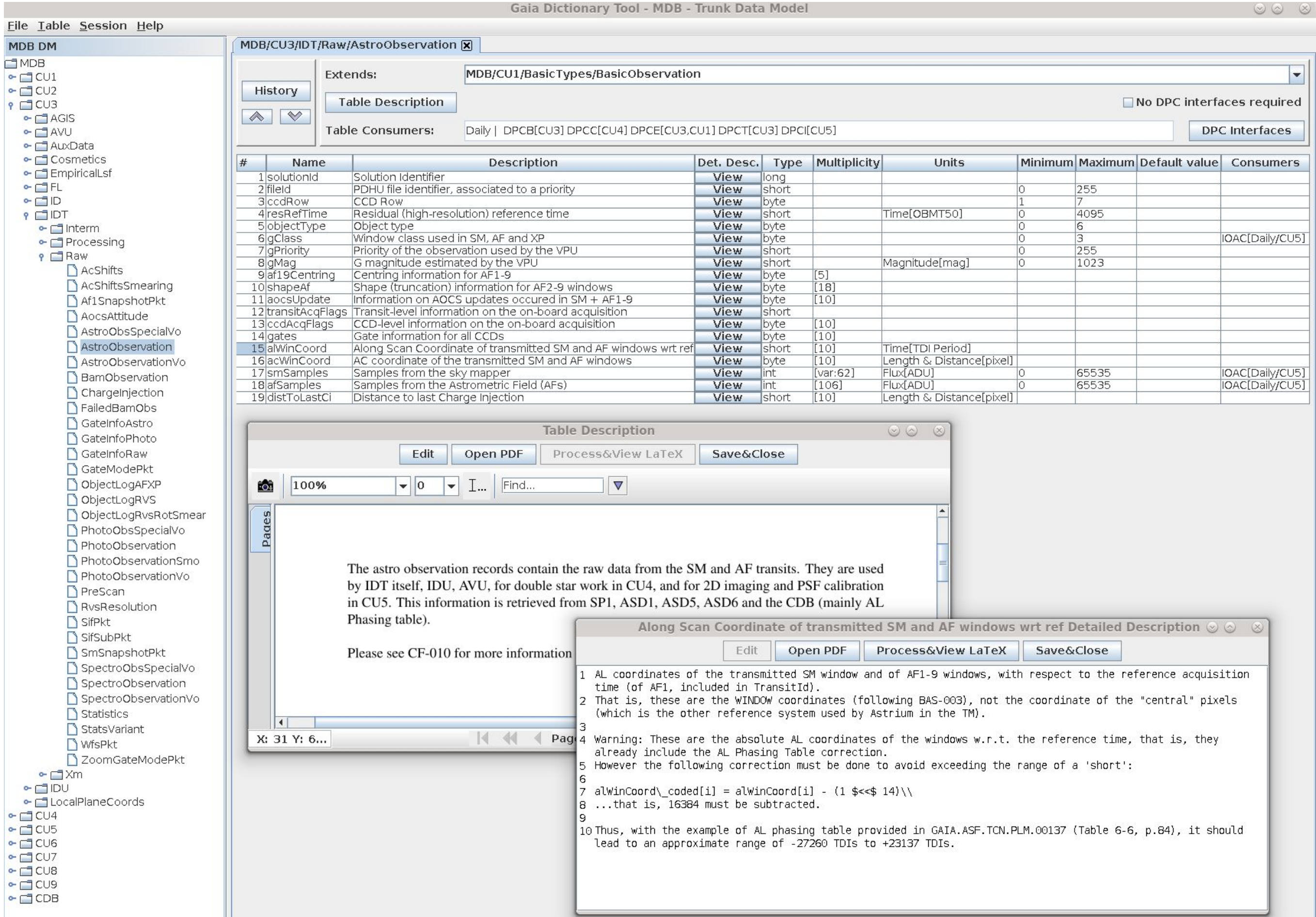
- Define/recommend the main programming language
  - Good for reusing "general" tools between units (algorithms, data access…)
  - Otherwise: code duplication in different languages, wrappers (overhead), …
- Test-oriented development
  - Define manageable Validation DataSets for continuous regression testing
  - Ensure determinism: beware of multithreading/parallelism race conditions
- Cyclic development
  - **Define *essential* and realistic features, implement first version ASAP**
  - Improve with "not-so-essential" ones progressively
- Implement realistic simulators ASAP
  - In Gaia we started it ~14 years before launch
  - Universe + Instrument models; data simulation at different levels
- Define an adequate **file format**
  - If possible, fulfilling (1) data processing in a center, (2) data exchange between centers, and even (3) bulk data publication
  - Nowadays, Parquet looks great for this
  - If you really have to use more formats, make sure that bidirectional conversion can easily and reliably be done

Technical considerations (II)

- Choose the right computing framework/approach, incl. I/O approach
  - E.g. in Gaia, huge number of records
    → a *database* may not be the right approach for number crunching
    ⇒ processing based on *files* (with some supporting DB for metadata)
  - But if you need a DB, choose *very* carefully the right approach for you. Nowadays, *columnar* DBs perform great - even for public/massive archives
- Ensure consistency in data, interfaces and software
  - Traceability + reproducibility
  - Tag and track *versions* of solutions, data model, software…
  - Consider adding a "Solution Identifier" to each and every table/file
- Define and clearly inform/describe central repositories for:
  - Software
  - Reference/test data
  - Working + released documents. BTW, LaTeX is *really* great.
  - If all these are organized following the Units and Centers, much better
  - Have a central point (wiki? Confluence?) with a clear list and basic description of all these services

Behold, the Holy Grail

- This is our 8th Wonder. Please, use or implement some tool to clearly **define and document your data model** in a centralized manner:
  - Systems, tables, fields, units, multiplicity, descriptions…
  - Transfers: consumers, periodicity
  - Size estimation
  - Reports
  - Automatic code
  - Sync with SVN/Git
- Seriously, this is one of the best things done in DPAC



Technical considerations (IV)

- Evaluate properly the **long-term** support (and license approach) of your language and tools
  - E.g.: JVM/JDK version (and vendor: Oracle, ehem…), Python version
  - Apache Commons, Numpy, Astropy, Pandas, …
- Also for project/software management tools, e.g.:
  - Code repository: SVN, Git
  - Issue tracking: We started with Mantis, migrated to JIRA later
  - Avoid, as much as possible, having to migrate during (or close to) operations
- Clearly define, *beforehand* or *ASAP*, a **software licensing approach** for all partners involved (ESA, institutes, universities, contractors…)
  - Much, MUCH difficult if done later
  - We're still struggling with it
- Define project services, shared calendars, mailing lists…
  - Create document templates for the various types (did I mention LaTeX already?)
  - Clearly define a "document codes" approach, and create (and update) a list of authors (+initials) and institutions
  - Create a list of acronyms ASAP and regularly maintain it
  - Create a Parameters Database. Now.

The Human dimension

- Prepare "**welcome packs**" and training resources
  - General project info including project services, repositories, essential code tools (e.g. basic guidelines to setup the code environment), document guidelines…
  - Coordinator/manager lists
  - What are you supposed to do?
    Who should you report to?
    Are you responsible (or will you be) for somebody else?
    Who will you work with (locally or remotely)?
    Does your work overlap with someone else's?
    Who can you ask when you get blocked with your subject?
- Don't postpone too much your tasks
  - *One JIRA a day keeps QA away*
- Accept the truth: **Some key people will leave the project at the worst time**
  - Be ready to replace him/her
  - Add redundancy before it's too late: **distribute the knowledge**
  - **Avoid single-person projects**: train and delegate tasks
- (Astro)physicists are not (software)engineers, and vice versa
  - You will probably need "translators" in between. Please have patience!
  - Accept that some excellent scientists may not program well → support them

In a nutshell

You're already late

In a nutshell

You’re already late

Don’t desperate, prioritize

In a nutshell

You're already late

Don't desperate, prioritize

Have as much fun as possible

In a nutshell

You're already late

Don't desperate, prioritize

Have as much fun as possible
(so you'll start from the lowest-priority task)

Thank you

Finançat per

Unió Europea
Fons Europeu
Next Generation

GOBIERNO DE ESPAÑA

Plan de Recuperación, Transformación y Resiliencia

Next Generation Catalunya

Generalitat de Catalunya