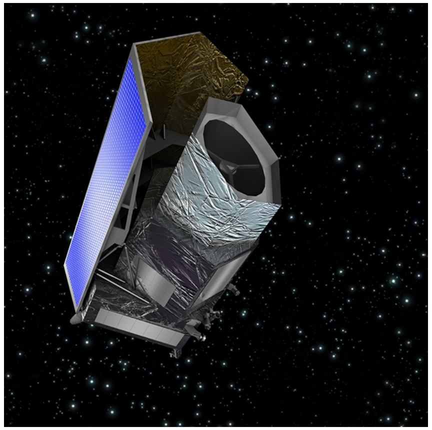
The Euclid ecosystem for software development and data processing

Martin Kümmel



Content

- Motivation
- Introduction to Euclid
- Software Development
- Processing
- What’s in for you?

Motivation

- Euclid is a **BIG** project!
- Astronomy/science evolves towards **BIG** projects:

  2MASS → **SDSS** → **BOSS** → **DES** → **Euclid** → **RUBIN/LSST** → ????
- What is **BIG**: Euclid Consortium > 1000 scientists, 100 developer;
- **BIG** projects have common (**BIG**?) problems!
- **BIG** projects find similar solutions!
- You might find items or even solutions presented here in your current/next **BIG** project!

Motivation (cont.)

Even if you never find yourself in a **BIG** project:

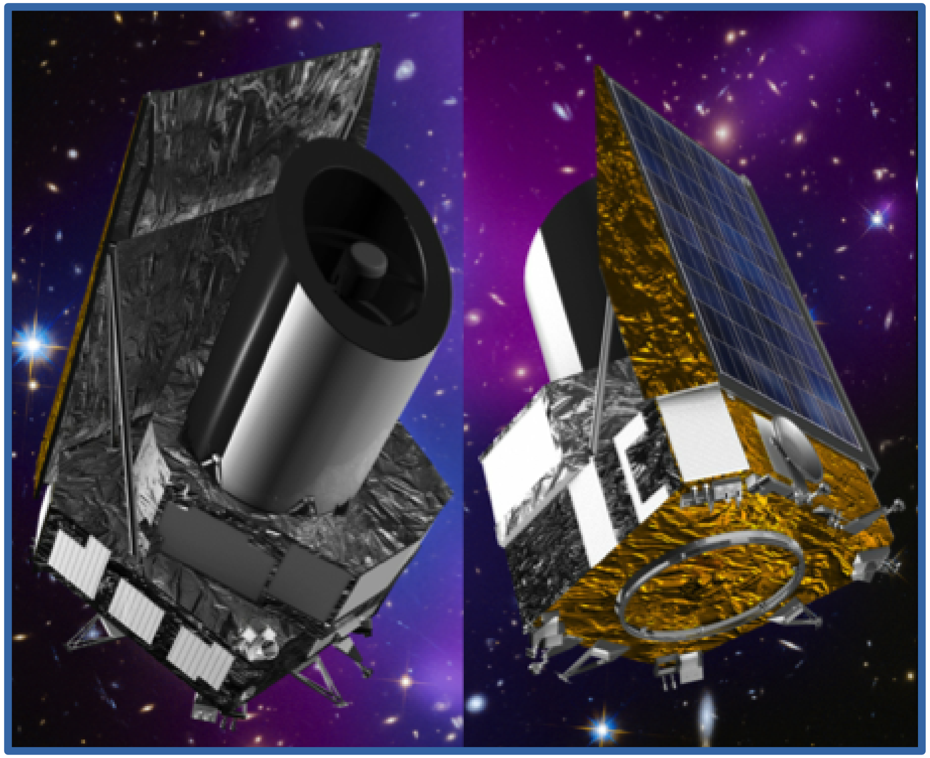
- Some Euclid solutions might be interested for:
  - Your project (master/PhD);
  - Your work;
  - Your group;

The Euclid satellite

Telescope	1.2m Korsch, 3 mirror anastigmat., f=24.5m				
Instrument	VIS	NIR			
Field-of-View	0.787x0.709 deg^2	0.763x0.722 deg^2			
Capability	Visual Imaging	NIR Imaging Photometry			NIR Spectroscopy
Wavelength range	550-900 nm	Y (920-1146 nm)	J (1146-1372 nm)	H (1372-2000 nm)	1100-2000 nm
Detector Technology	36 arrays 4k x 4k CCD	16 arrays 2k x 2k NIR sensitive HgCdTe detectors			
Pixel Size/FWHM	0.1" / 0.2”	0.3 " / 0.3”			0.3 "/
Spectr. Res.	-	-			R=250

Euclid Mission

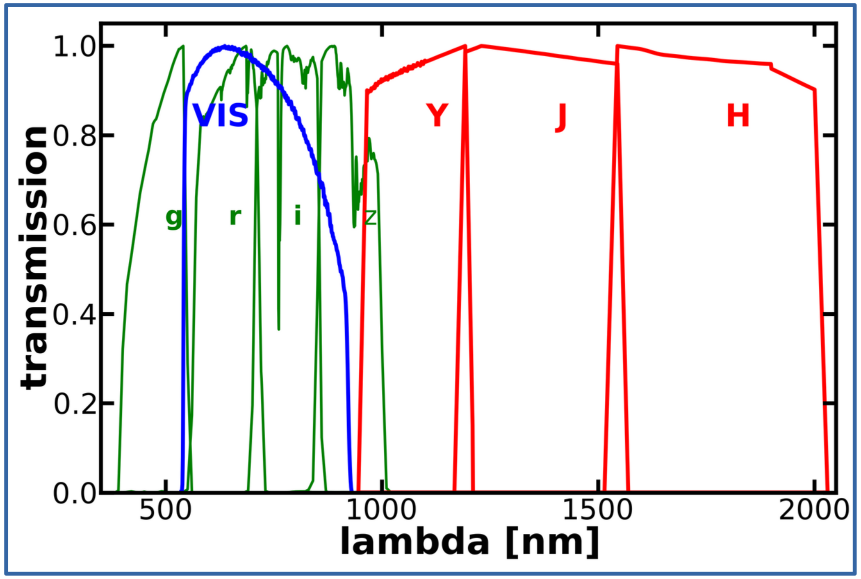
- 1.2-m Korsch telescope
- VIS instrument 550-900nm
- NISP instrument:
  - photometry in Y, J, H
  - slitless spectroscopy Δλ/λ ~250
- FOV: 0.54deg^2 VIS+NISP
- 15,000 deg^2 wide survey
- 40 deg^2 deep survey
- External data from ground based surveys (for photo-z):
  - DES
  - LSST
  - Pan-STARRS
  - UNIONS



Launch: June 2024

Euclid Science

- Determine the expansion of the Universe at various cosmic ages
- **Weak lensing analysis**
- **Galaxy clustering**
- Redshifts (griz) are required:
  - From NISP slitless spectroscopy
  - From photo-z
- **Legacy science!**



Software development: environment

- “Miracle software” phenomenon: you get ***best software ever***, but you cant get it installed…
- → a stable, robust and reproducible environment must be provided to:
  - Facilitate SW development for many developers;
  - Do all necessary testing;
  - Run the code on different scales (up to production);
- The Euclid solution: EDEN (**E**uclid **D**evelopment **En**vironment)
  - common standards (O/S, libs, …);
  - rules (coding standards, Software Engineering guide, …)
  - tools (compilers, packaging, …)
- Distributed as:
  - Virtual Machine;
  - Docker container;

Software development: libs/cvmfs

- EDEN-3.0 system: **CentOS7.9/gcc-c++ 9.3.0/python3.9.9**
- EDEN-3.0 libs: DEMO (https://euclid.roe.ac.uk/projects/codeen-users/wiki/EDEN_v30)
- Central in EDEN-3.0: **C**ern **VM** **F**ile **S**ystem (https://cernvm.cern.ch/fs/):
  - software distribution service;
  - server/client solution;
  - software and libs **centrally** installed on the server
  - software and libs **locally** mounted by client at "/cvmfs"
  - environments are set accordingly;
  - → **system is automatically updated!**
  - DEMO (cvmfs file system)

Software development: make/install

- Central for Software Development: **Elements** (https://github.com/astrorama/Elements)
- Elements does:
  - Build **`python`**/**`C++`** executables and libraries;
  - Package the various **`python`**/**`C++`** projects with management of their dependencies
- Elements:
  - Came from **CERN** (I think…);
  - Is based on **`CMAKE`**;
  - Supports version specific dependencies;
  - Hierarchical search path for dependencies;
  - Offers helper scripts to create software stubs;
  - Contains executables to run software!
  - DEMO Elements;

Software development: deployment

- Software storage: privately hosted gitlab repository (https://gitlab.euclid-sgs.uk/)
- Development model:
  - Main branch “develop”;
  - Active development on feature branches;
  - Version tagging from “develop”;
  - Numbering scheme for tags;
  - DEMO Euclid development model;
- SW deployment Jenkins:
  - Continuous integration via automatic:
    - Code update;
    - Code compilation;
    - Code deployment to cvmfs;
    - Version control!
  - DEMO Jenkins

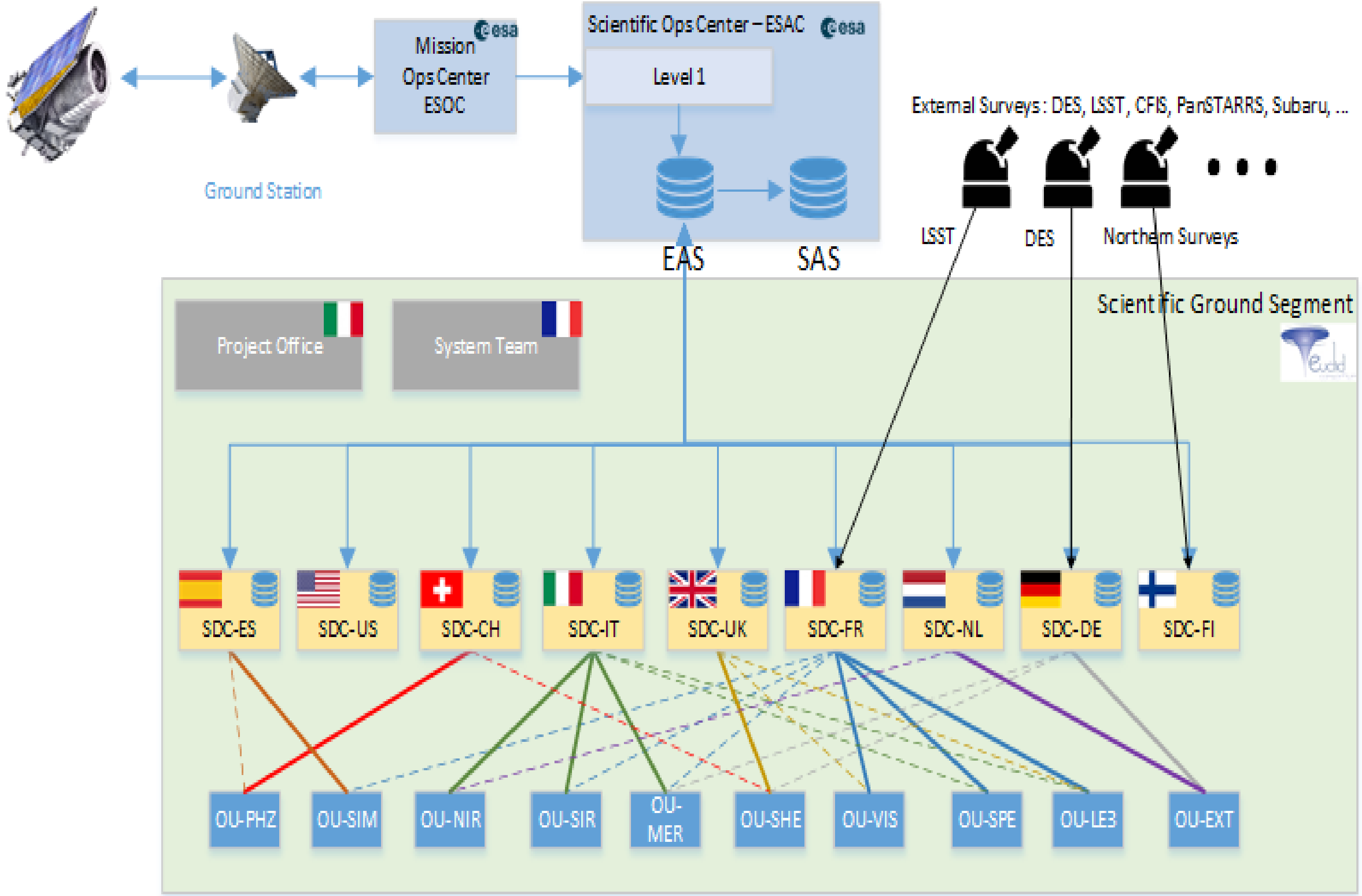
Software development: quality control

- SonarQube (https://www.sonarqube.org/);
- Maturity levels;
- Maturity assessments;
- DEMO (https://euclid.roe.ac.uk/projects/quality-tools/wiki/Wiki);

Processing

- Data model:
  - Definition of the entities;
  - Definition of the products;
- Archive:
  - **EAS**: Euclid Archive system;
  - Used for processing and storage!
  - Data model is mapped onto the **EAS**!
- Processing:
  - In the Science Data Centers (**SDC**);
  - One **SDC** per country (~9 in total);
  - ~5000 cores per **SDC**;
  - **SDC**'s are inhomogeneous (dedicated vs. general purpose);
  - No GPU's;

Processing overview



What is in for you!

- For everyone: git/svn/cvs!
- For everyone: choose a good build system (maybe Elements?);
- For PhD project and larger: automatic compiling and testing (Jenkins);
- For PhD project and larger: develop/run in a **VM** or **container**;
- For larger projects (from group level on): archive support;
- For big projects:
  - Use **`cvmfs`** or similar systems;
  - Code inspection and maturity levels;
  - Archive driven processing;