



# DATA QUALITY COMMON TOOLS

— Stefano Cavuoti —  
On Behalf of DQCT Team  
with the contribution of EAS Team

---

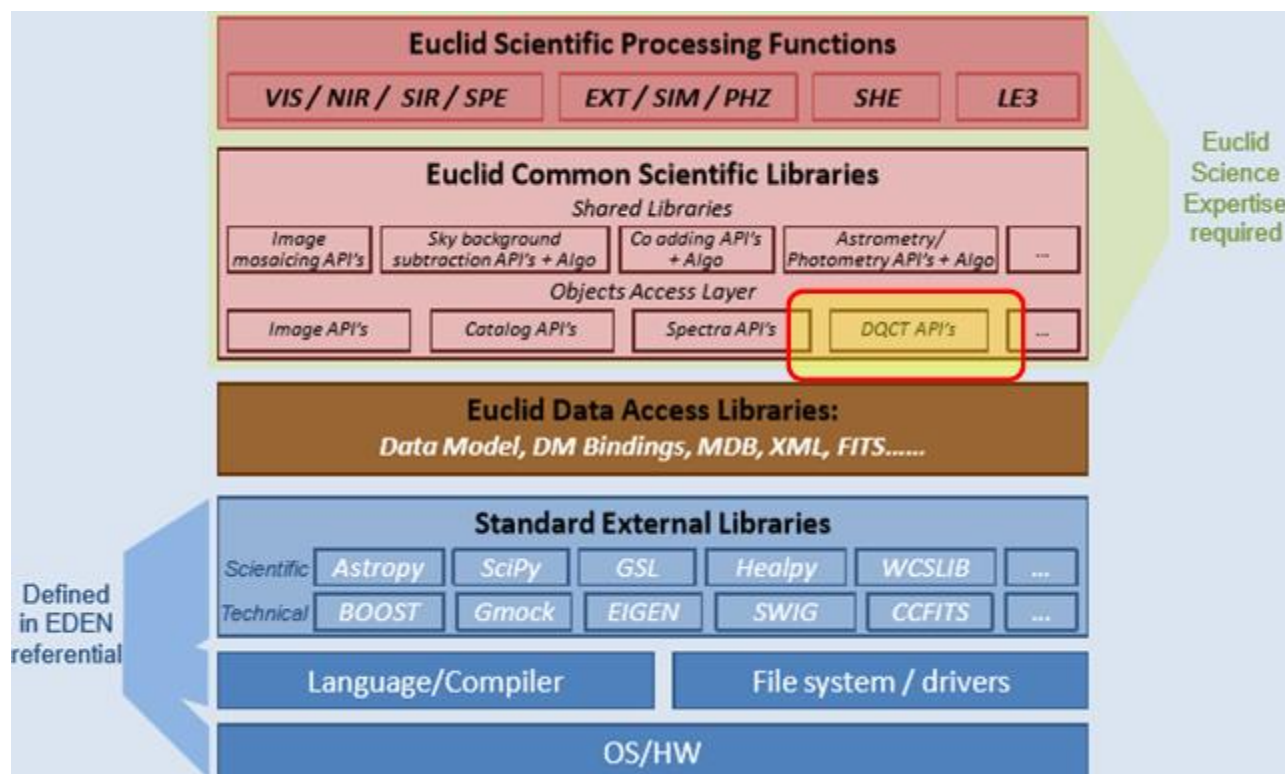
---

# DQCT COMMITMENTS

- DQCTs deal with **common tools** concerning quality flags, error estimates, statistics like mean, standard deviation, RMS, S/N etc., as well as any other metadata parameter produced by the pipeline, collected in quality reports specific to each data level and stored on EAS during pipeline processing;
- DQCTs will provide tools to produce visual/graphical inspection products, such as thumbnails, trend analysis diagrams, histograms, scatter plots, etc. These will be also used by QualityWISE, a data quality visualization system provided by EAS Team;
- Despite of its nature of “common” tools, specific DQ tools **may** be provided for particular types of processing;
- The DQCTs will be integrated into the pipelines (as libraries/API) and used by pipeline developers;
- DQCTs are and must be based on specifications and requirements agreed with OUs. Then we will harmonize them, by finally developing and providing such SW tools to SDC-DEVs for their integration within pipelines;
- ★ **DQCTs are NOT in charge to evaluate and assess the quality of Euclid scientific data, but to provide standard (common) software tools to check and register the data quality of each scientific product.**

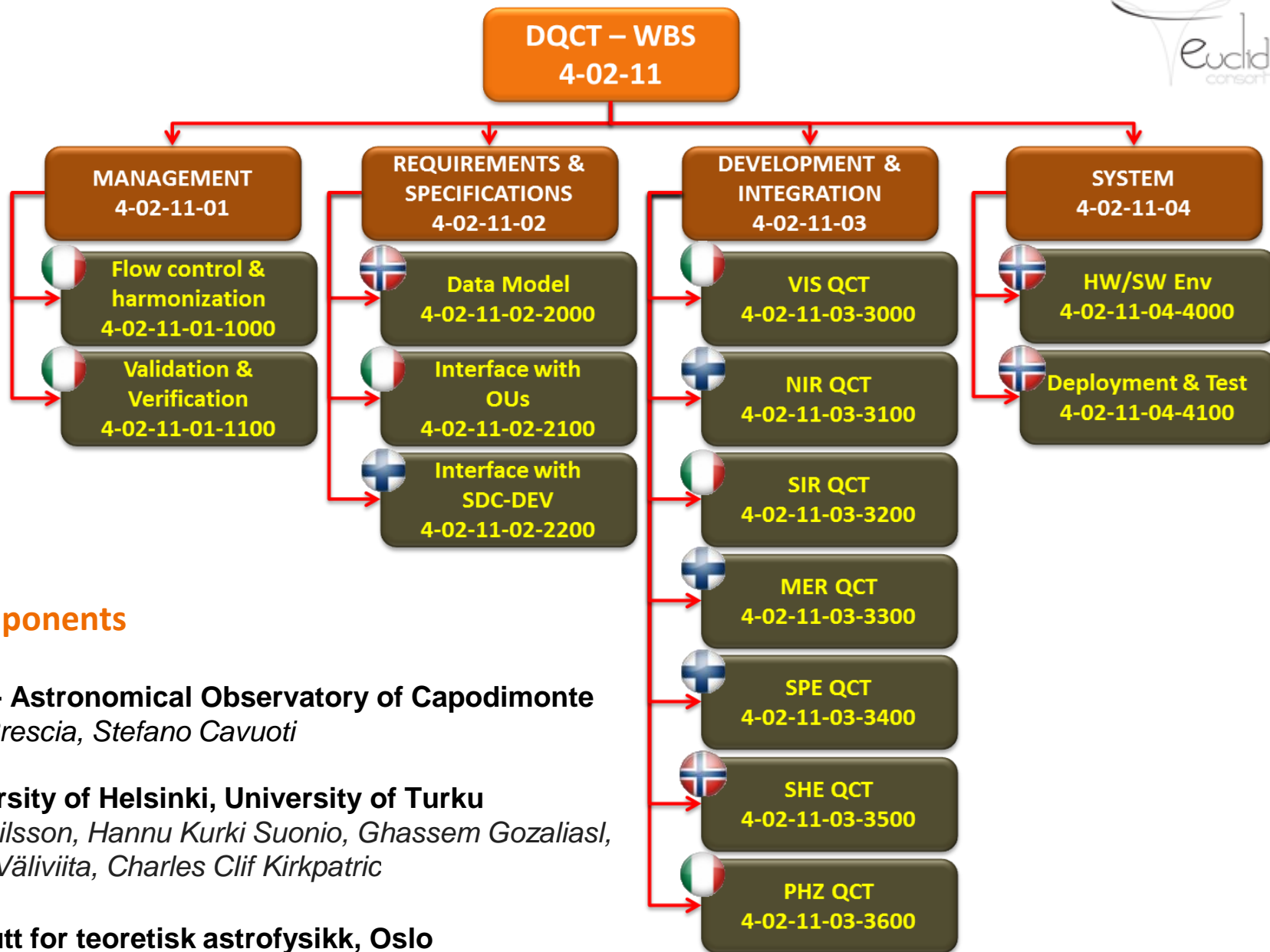
# DQCT Main Interfaces

- ★ Gather, homogenize and harmonize algorithmic and scientific requirements from all OUs;
- ★ Formalize and harmonize data model and standards in collaboration with OUs and STs;
- ★ Autonomous DQCT framework, to minimize interaction with SDC-DEV and EAS during software production and debugging, and to facilitate a rapid development-test loop;
- ★ Simulated data used to test DQCTs;
- ★ Interact with SDC-DEV teams to integrate and test DQCTs on all SGS pipelines;



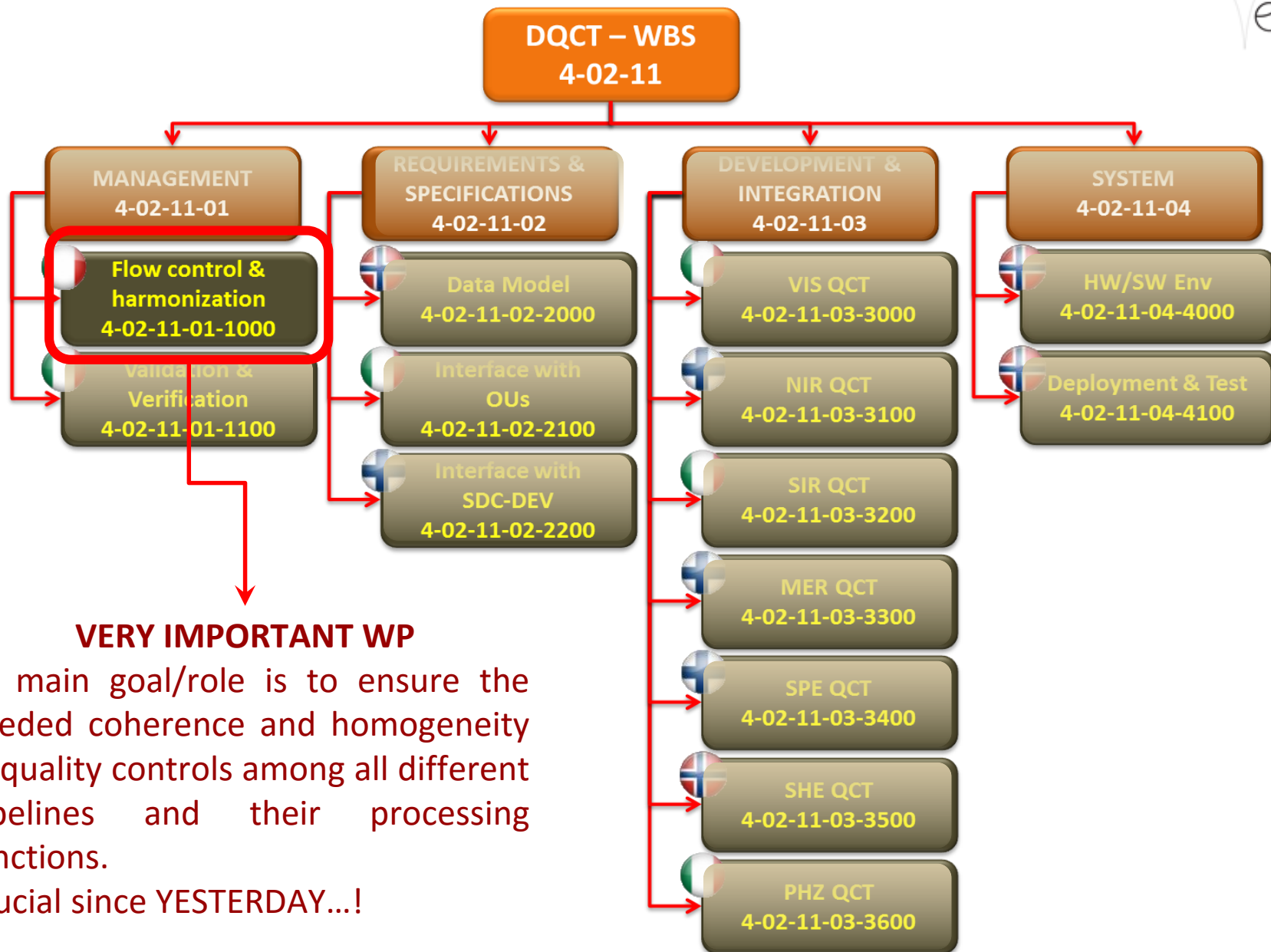
# DQCT Main Interfaces

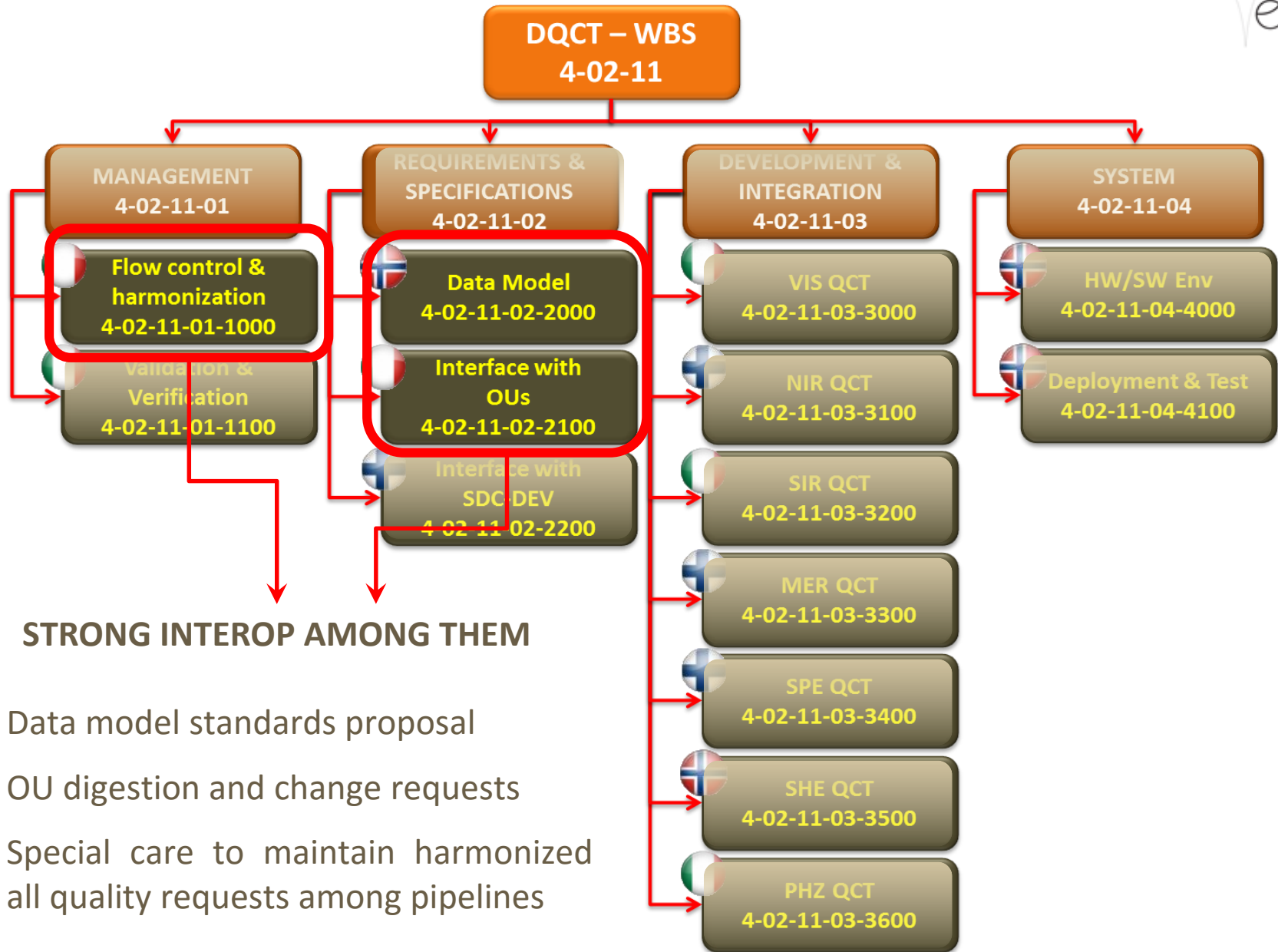
- ★ To provide "common" data quality tools at all data levels except L1;
- ★ To ensure that the interface of quality and HK information coming from L1 processing is compatible with what we expect;
- ★ The most important delivery is an incremental quality report associated to each data product, to be built and propagated through the pipelines and stored within the EAS;
- ★ The "common" tools will be integrated and used within the pipelines to avoid any potential delay during the data reduction, due to off-line calculations;
- ★ DQCT WP will provide Application Programming Interfaces (APIs) in the form of C/Python software packages, to be nested within the pipeline code at all SGS data flow levels in which quality checks are required.



## WP Components

-  **INAF - Astronomical Observatory of Capodimonte**  
*Max Brescia, Stefano Cavuoti*
-  **University of Helsinki, University of Turku**  
*Kari Nilsson, Hannu Kurki Suonio, Ghassem Gozaliasl, Jussi Väliiviita, Charles Clif Kirkpatric*
-  **Institutt for teoretisk astrofysikk, Oslo**  
*Stein Vidar Hagfors Haugan, Martin Wiesmann*





- Data model standards proposal
- OU digestion and change requests
- Special care to maintain harmonized all quality requests among pipelines

# Data Quality Report Design

**Quality reports (QRs) must be attached to all relevant data products. QRs are incremental as data are propagated through pipelines, with the minimum amount of redundancy.**

The idea is to ensure that quality information directly related to a sub-product is available "near" or alongside the sub-product in terms of placement in the overall Data Model, while preventing duplication of quality information from progenitors.

Nevertheless, progenitor and descendant information - both quality information and "the actual data" - must be accessible by querying the DB.

For more details:

<http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/Dqreport>



# DQCT DATA MODEL DESIGN



**The DQCT DM proposes “common” solutions for quality controls/flags, to be followed by (and decided with) OUs and SDCs within their pipelines.** According to this approach the location of DQCT DM is into the **dictionary/bas** branch. Due to slow feedback from OUs, the OU-specific part of the DQCT DM is currently based only on "suggestions" from the DQCT team. Its current status (in the SVN repository) can be regarded more or less like a placeholder.

The DQCT DM is logically separated into two parts, one concerning specific data quality parameters for each OU, and one concerning the dynamic flag system.

## **The VIS-DQCT lesson**

Through a careful analysis of the VIS quality requirements, a DM proposal was placed in a wiki page, but not yet implemented in the SVN. After an interaction with OU-VIS we decided to update the VIS DM in SVN to reflect the current consensus - although future adjustments will occur.

**We hope that when our DM section related to VIS will be updated, it will be included/imported into the VIS DM branch itself. The same strategy will be used w.r.t. other OUs.**

With respect to the dynamic flag system, the DM is more than a placeholder, but certainly not final. Work will be done in collaboration with RUG to harmonise the DQCT dynamic flag system with the quality viewer (to be based on QualityWISE).

# Interaction with OUs - First Approach

Current Data Level	Parameter Name	Meaning	Input Needed
1	S/N	Signal-to-noise of raw image	Raw Image
1	Flag x	OK/NOT OK	Raw Image
2	Flag y2	0 high 1 medium 2 low	Flag x1 Flag x1
2	...	...	...
3	Flag w3	OK/NOT OK	Flag y2 Flag x1

**initially proposed to OUs asking for a feedback...**

Several proposals and requests sent to all OUs to gather DQ information to be produced by DQCT and integrated within SDC pipelines...practically wasted time!



# Second Approach

1. Derive a draft for each pipeline
2. Submit the draft to OUs
3. Get Feedbacks
4. Goto 1

If the mountain will not come to Mahomet,  
Mahomet will go to the mountain



## CONs:

**This approach requires a surplus of work  
and time by both sides**

**More iterations are required**

**PRO: .....**



# Second Approach

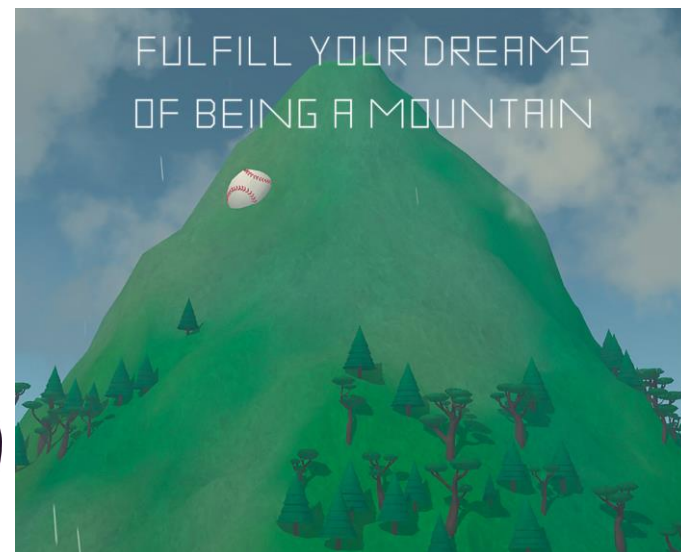
1. Derive a draft for each pipeline
2. Submit the draft to OUs
3. Get Feedbacks
4. Goto 1

## CONS:

**This approach requires a surplus of work and time by both sides**

**More iterations are required**

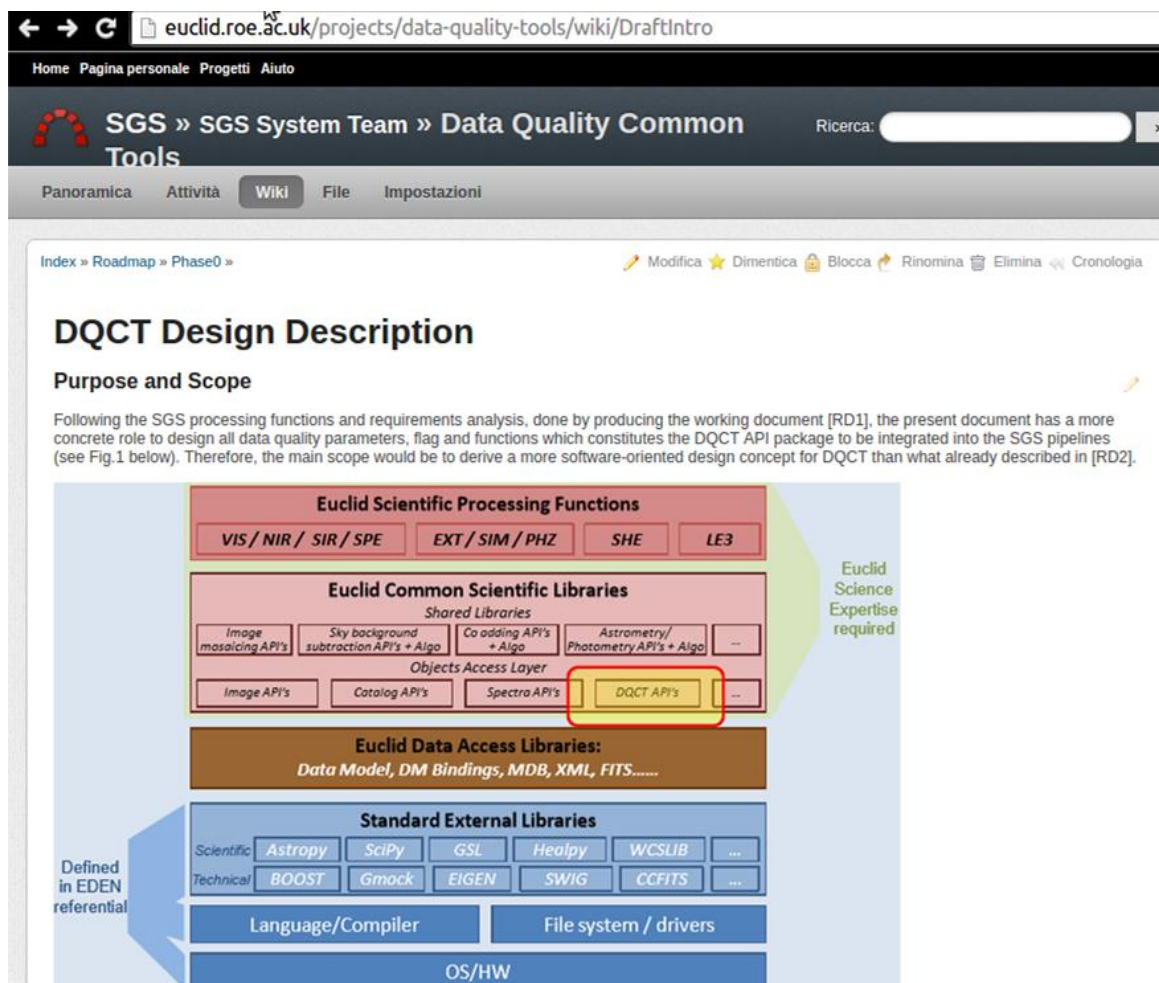
**PRO: .....IT WORKS!!!**



# DQCT pipeline drafts

The proposals of DQCT products are available on redmine:

<http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/DraftIntro>



The screenshot shows a web browser window with the URL [euclid.roe.ac.uk/projects/data-quality-tools/wiki/DraftIntro](http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/DraftIntro). The page title is "DQCT Design Description" and the sub-section is "Purpose and Scope". The text describes the role of the DQCT API package in the SGS pipelines. Below the text is a diagram of the software stack architecture.

**Fig.1 The DQCT API component within the SGS software stack architecture [RD3]**

The diagram illustrates the software stack architecture, showing the DQCT API component within the Euclid Common Scientific Libraries. The stack is organized into several layers:

- Euclid Scientific Processing Functions:** Includes VIS / NIR / SIR / SPE, EXT / SIM / PHZ, SHE, and LE3.
- Euclid Common Scientific Libraries:**
  - Shared Libraries:** Image mosaicing API's, Sky background subtraction API's + Algo, Co adding API's + Algo, Astrometry/Photometry API's + Algo, ...
  - Objects Access Layer:** Image API's, Catalog API's, Spectra API's, **DQCT API's** (highlighted with a red box), ...
- Euclid Data Access Libraries:** Data Model, DM Bindings, MDB, XML, FITS.....
- Standard External Libraries:**
  - Scientific:** Astropy, SciPy, GSL, Healpy, WCSLIB, ...
  - Technical:** BOOST, Gmock, EIGEN, SWIG, CCFITS, ...
- Language/Compiler** and **File system / drivers**
- OS/HW**

A green arrow on the right side of the diagram points to the "Euclid Science Expertise required" label. A blue arrow on the left side points to the "Defined in EDEN referential" label.

## Current Index

1. [VIS Pipeline DQCTs](#)
2. [NIR Pipeline DQCTs](#)
3. [SIR Pipeline DQCTs](#)
4. [Quality Report Design](#)
5. [Interface with QualityWISE](#)

other pipelines coming next...

# What we are looking for

For each step of the pipelines we need to find:

Parameters

Quality Control Functions

Quality flags

Trend analyses

Data Model Specifications (if any)

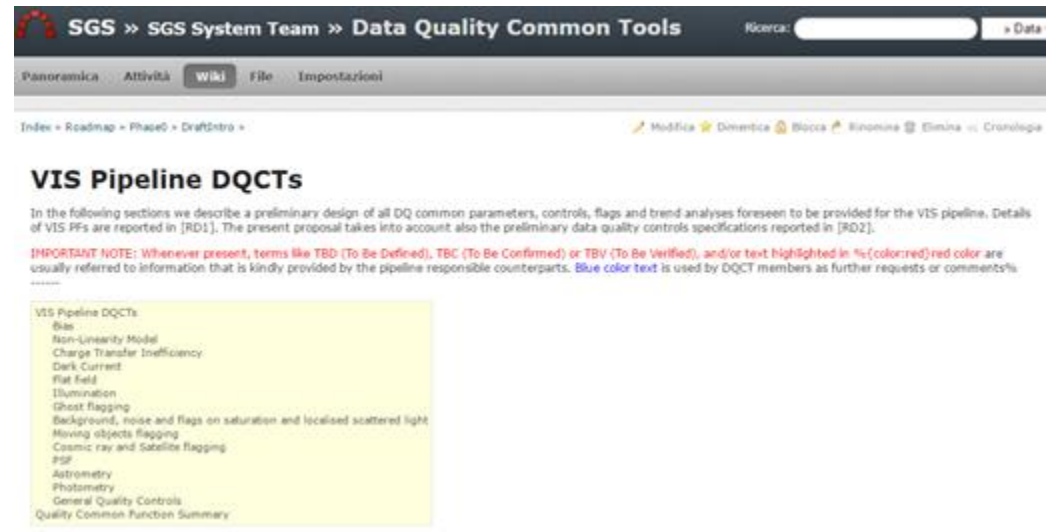


...and they must be found for each pipeline

# Pipeline Sections - VIS Example

QC tools for:

Bias  
Non-Linearity Model  
CTI  
Dark Current  
Flat field  
Illumination  
Ghost flagging  
Background and saturation  
Moving objects flagging  
Cosmic rays  
PSF  
Astrometry  
Photometry  
General Quality Controls



The screenshot shows a web browser interface for the 'Data Quality Common Tools' project. The page title is 'VIS Pipeline DQCTs'. The content includes an introductory paragraph and an important note about terminology. A yellow box highlights a list of pipeline sections:

- Bias
- Non-Linearity Model
- Charge Transfer Inefficiency
- Dark Current
- Flat field
- Illumination
- Ghost flagging
- Background, noise and flags on saturation and localised scattered light
- Moving objects flagging
- Cosmic ray and satellite flagging
- PSF
- Astrometry
- Photometry
- General Quality Controls
- Quality Common Function Summary

For more details: <http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/Visdqct>



# Parameters - Bias Example



## Raw Bias

**BiasLevel** - average bias level (mean of the bias pixel values);

**BiasPreScanLevel** - average pre-scan region bias level (mean of the bias pixel values);

**BiasOverScanLevel** - average over-scan region bias level (mean of the bias pixel values);

**BiasMode** - sample mode of the bias pixel values;

**BiasPreScanMode** - sample mode of the pre-scan region bias pixel values;

**BiasOverScanMode** - sample mode of the over-scan region bias pixel values;

**BiasStdev** - sample standard deviation of the bias pixel values;

**BiasPreScanStdev** - sample standard deviation of the pre-scan bias pixel values;

**BiasOverScanStdev** - sample standard deviation of the over-scan bias pixel values;

...

## Master Bias

**MasterBiasLevel** - mean value of the master bias levels;

**MasterBiasPreScanLevel** - average pre-scan region master bias level (mean of the bias pixel values);

**MasterBiasOverScanLevel** - average over-scan region master bias level (mean of the bias pixel values);

**MasterBiasMode** - mode value of the master bias levels;

**MasterBiasPreScanMode** - sample mode of the pre-scan region master bias pixel values;

**MasterBiasOverScanMode** - sample mode of the over-scan region master bias pixel values;

**MasterBiasStdev** - sample standard deviation value of the master bias levels;

...



# QC Functions - Bias Example



Taking into account the parameter listed above, these are the functions to be implemented:

## ***StDev(array)***

It calculates the standard deviation for an input array (1D or 2D)  
It can be used to calculate the parameter BiasStdDev.

## ***Mean(array)***

It calculates the mean for an input array (1D or 2D)  
It can be used to calculate the parameter BiasLevel.

## ***Mode(array, binsize)***

It calculates the mode for an input array (1D or 2D) with a given binsize  
It can be used to calculate the parameter BiasMode.

## ***MinMax(array)***

It calculates the minimum and the maximum of an input array (1D or 2D)  
It can be used to calculate the parameters MinBias, MaxBias, MasterBiasLevel, MasterBiasStdev, MasterBiasStdevDifference and MasterBiasMaxSubwinStdev. It also is a service routine used in the calculation of BiasFlatness and MasterBiasSubwinFlatness parameters.

## ***Median(array)***

It calculates the median of an input array (1D or 2D)  
It can be used to calculate the parameters BiasMedianLevel and MasterBiasMedianLevel. It also is a service routine used in the calculation of BiasFlatness and MasterBiasSubwinFlatness parameters.

## ***Flatness(array[array])***

It calculates the difference between minimum median and maximum median of an array of sub-windows or images. It makes use of MinMax(array) and Median(array) functions.  
It can be used to calculate the parameter BiasFlatness.

## ***SigmaClip(array, base, threshold)***

It performs the SigmaClipping on an array, based on a specific "base" (mean, median or a generic value) and a rejection threshold ( $1\sigma$  or higher). It can be used to perform the foreseen master bias SigmaClipping and to store the base and threshold used.

## ***FourierAnalysis(array)***

It performs a Fourier Analysis, providing as output the frequency and the significance of the frequency.

# Flags - Bias Example

## Raw Bias

**BiasStdevExceeded** – standard deviation over the XX threshold;

**BiasLevelExceeded** - bias level over the XX threshold;

**BiasModeExceeded** - bias mode over the XX threshold;

**BiasStdevLightLeakingExceeded** – stdev over the XX threshold between pre- and over- scan;

**BiasLevelLightLeakingExceeded** - bias level over XX threshold between pre- and over- scan;

**BiasModeLightLeakingExceeded** - bias mode over XX threshold between pre- and over- scan;

**BiasFlatnessExceeded** - bias flatness over the XX threshold. For background uniformity;

**BiasPowerFreqExceeded** - significance of frequency detected over the XX threshold;

## Master Bias

**MasterBiasStdevExceeded** – master bias standard deviation over the XX threshold;

**MasterBiasLevelExceeded** – master bias level over the XX threshold;

**MasterBiasModeExceeded** – master bias mode over the XX threshold;

**MasterBiasStdevLightLeakingExceeded** – stdev over threshold between pre- and over- scan;

**MasterBiasLevelLightLeakingExceeded** - level over threshold between pre- and over- scan;

**MasterBiasModeLightLeakingExceeded** - mode over threshold between pre- and over- scan;

**MasterBiasFlatnessExceeded** – flatness over threshold. For bias uniformity;

**BiasRMSDiffExceeded** – abs difference between two bias stdevs over threshold;

# Trends - Bias Example

This section describes a preliminary list of the trend analyses that could be performed in the specific case.

## **Raw Bias**

**percentile trend of standard deviation for N bias values;**

**Analysis of deviation within  $XX\sigma$  of N past bias values;**

## **Master Bias**

**percentile trend of standard deviation for N master bias values;**

**Analysis of deviation within  $XX\sigma$  of N past master bias values;**

# Data Model - Bias Example

This section is dedicated to specify all entries foreseen for the DQCT data model branch (dictionary/bas) for the specific subject. A consistency check and update is required at future stages.

**NOTE for data model: each raw bias statistic is needed per sub-areas of a quadrant, that in the data model can be structures as a single array.**

# DQCT Status

## VIS Pipeline: **advanced**

Our proposal has been reviewed by the OU-VIS people, minor further changes are foreseeable (we thank Olivier Herent)

<http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/Visdqct>

## SIR Pipeline: **started feedback on first draft**

Our draft has been submitted to the OU-SIR people, a first answer has been received few days ago. We also planned a f2f meeting in Milan in June (we thank Bianca Garilli)

<http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/Sirdqct>

## NIR Pipeline: **first draft deployed last week on the wiki**

We are finalizing the first draft and as soon as possible will be submitted to OU-NIR people

<http://euclid.roe.ac.uk/projects/data-quality-tools/wiki/Nirdqct>

# Harmonization among pipelines

After the design of DQCTs for the single pipelines, the important consequence is to find commonalities. For example, by finding common quality control functions to implement them as standard APIs.

## VIS

## SIR

FUNCTION NAME	WHERE REQUIRED
<i>ChiSquare(array)</i>	VIS (PSF, Astrometry)
<i>ChiSquareFit(array, deg)</i>	VIS (Illumination, Non Linearity)
<i>Flatness(array[array])</i>	VIS (Bias)
<i>FourierAnalysis(array)</i>	VIS (Bias)
<i>GetFlag(flagname)</i>	everywhere
<i>GetParameter(name)</i>	everywhere
<i>GetTrailingLevel()</i>	VIS (Flat)
<i>MapCombiner(array[array])</i>	VIS (General)
<i>MapCounter(array, threshold)</i>	VIS (Dark, Flat, Ghost, Scattered Light, Cosmic ray)
<i>Mean(array)</i>	VIS (Bias, Dark, Flat, Illumination)
<i>Median(array)</i>	VIS (Bias, Dark, Flat, Illumination)
<i>MinMax(array)</i>	VIS (Bias, Dark, Flat, Illumination, PSF, Non Linearity)
<i>Mode(array)</i>	VIS (Bias, Dark, Flat, Illumination)
<i>Percentiles(array, fraction)</i>	VIS (Dark, Flat, Illumination)
<i>PRNUEval(array, reference)</i>	VIS (Flat)
<i>PSFdetectionLimit(array)</i>	VIS (PSF)
<i>QEMap(array)</i>	VIS (Flat)
<i>RMS</i>	VIS (Non Linearity, Astrometry)
<i>SetFlag(flagname, value)</i>	everywhere
<i>SetParameter(name, value)</i>	everywhere
<i>SetTrailingLevel(value)</i>	VIS (Flat)
<i>SigmaClip(array, base, threshold)</i>	VIS (Bias, Dark, Flat, Scattered Light)
<i>StDev(array)</i>	VIS (Bias, Dark, Flat, Illumination, PSF)

FUNCTION NAME	WHERE REQUIRED
<i>GetTrailingLevel()</i>	SIR (Flat field)
<i>MapCombiner(array[array])</i>	SIR (Cosmic ray and moving objects flagging)
<i>MapCounter(array, threshold)</i>	SIR (Cosmic ray and moving objects flagging, Flat field, 2D Spectra, 1D Spectra)
<i>Mean(array)</i>	SIR (Flat field)
<i>Median(array)</i>	SIR (Flat field)
<i>MinMax(array)</i>	SIR (Flat field)
<i>Mode(array, binsize)</i>	SIR (Flat field)
<i>SetTrailingLevel(value)</i>	SIR (Flat field)
<i>SigmaClip(array, base, threshold)</i>	SIR (Flat field)
<i>StDev(array)</i>	SIR (Flat field)

+ NIR + ....



**common/specific QC parameters/functions/flags/trends**

# QualityWise - DQCT interaction



The Euclid Archive System (EAS) has the following requirement:  
R-EAS-M-072: EAS shall have a quality service which allows the user to check quality information for the selected data product.

This requirement has caused concern, since we clearly must avoid overlapping responsibilities between the EAS team and the Data Quality Common Tools (DQCT) team.

22nd September 2015 a dedicated videoconference was held to discuss this issue. Members of the EAS and DQCT teams participated

**Following slides report the conclusions of this videoconference**

# EAS Quality Service



The EAS Quality Service will consist of a data browser and data visualisation system to provide a convenient method of looking at quality information for products and their progenitors;

Will make use of lineage information in the EAS;

Quality information will not be generated. This is the job of the DQCT;

The QualityWISE browser is an example of such a system

As well as browsing, QualityWISE allows a single manual flag to be set and a comment to be entered.

For more details on QualityWISE see <http://arxiv.org/pdf/1203.4208v2>

QualityWISE was itself based on QualityFITS for MegaCAM

For more info on QualityFITS see <http://www.cfht.hawaii.edu/Science/CFHLS/T0007/T0007-docsu23.html>

The EAS Quality Service has been referred to before as a Quality Data Tool

The use of the word “tool” here has caused some confusion, since it incorrectly implies data is being modified.

# Conclusions and Actions



## Conclusions

All Quality tools will be generated by the DQCT WP (responsibility of the DQCT team). The *proposal-feedback* loop is the official strategy in progress;

The EAS Quality Service browses and visualizes data and will be under responsibility of the EAS team;

We need to define the aspects of EAS User Interfaces in very broad terms and how the user can interact with them (calibration scientists, survey groups, instrument ops teams, quality, DP coordination, etc).

## Actions

The EAS team will provide a system resembling QualityWISE, working with the Euclid Data Model. The prototype will use OU-EXT data, since the data model is more mature.

Next interaction DQCT-EAUG to provide input on the type of quality data to be managed by QualityWISE

OU-EXT to review the current DQCT part of the data model (but all OUs should be doing this!)



**Thank you for your attention**