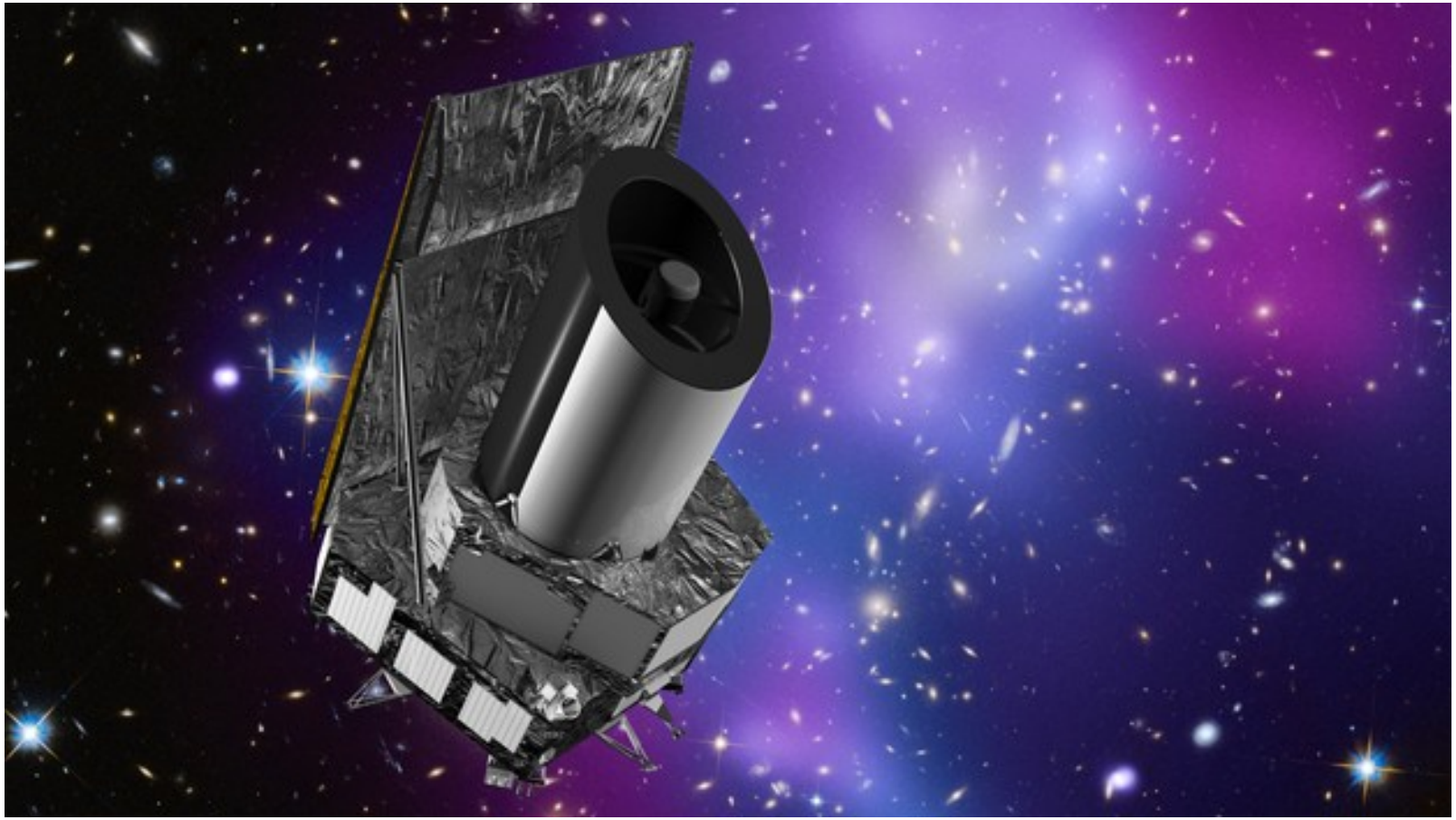
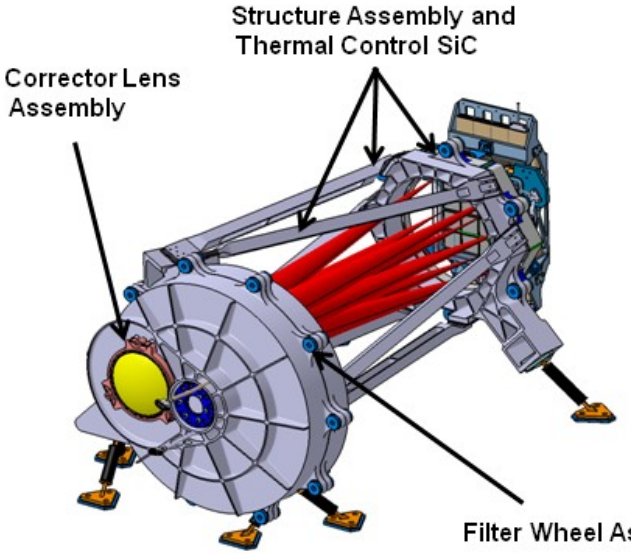
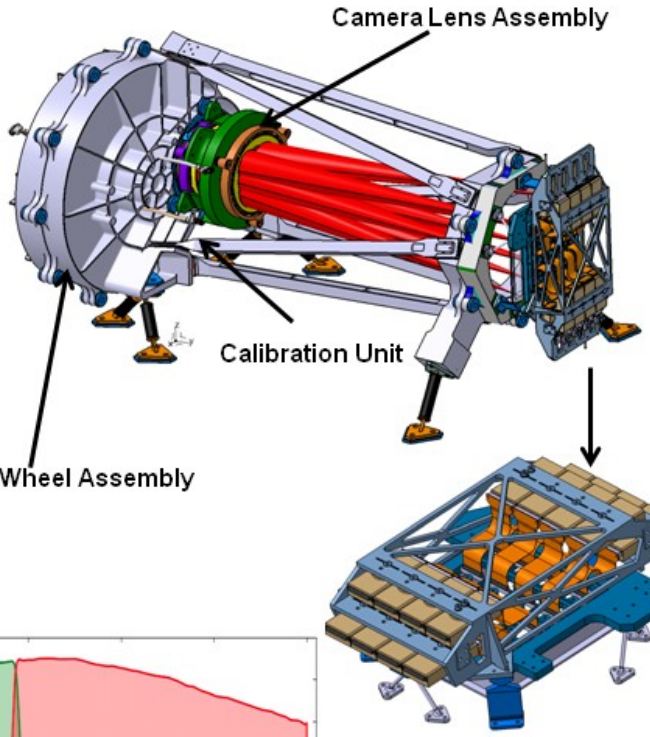
Projet EUCLID: Tests logiciels

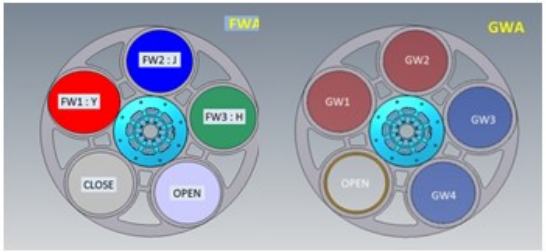
Sylvain Ferriol, IPNL



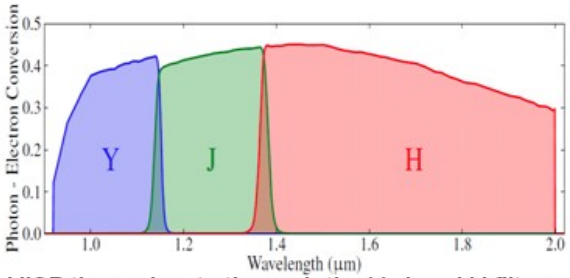
Acquisition pour la caractérisation des capteurs du Near IR Spectrometer Photometer (NISP)







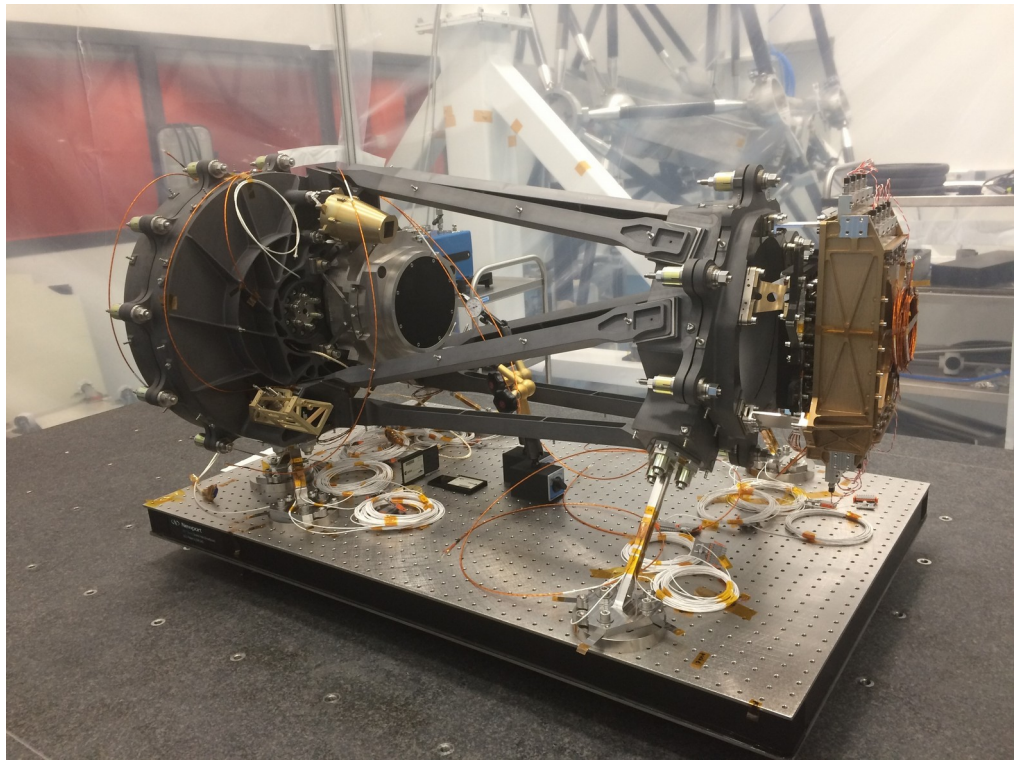
Filters and grisms positions in wheels

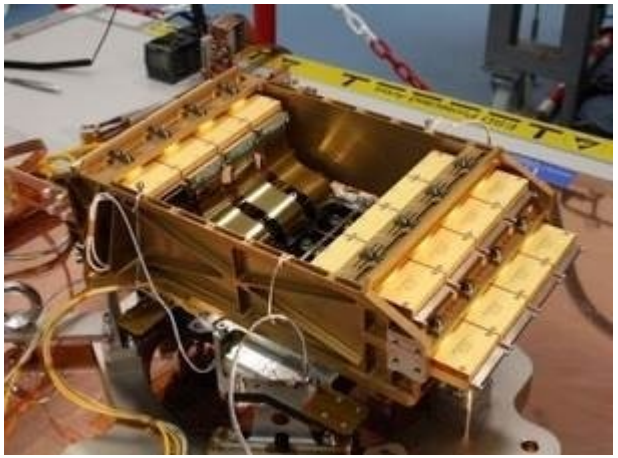


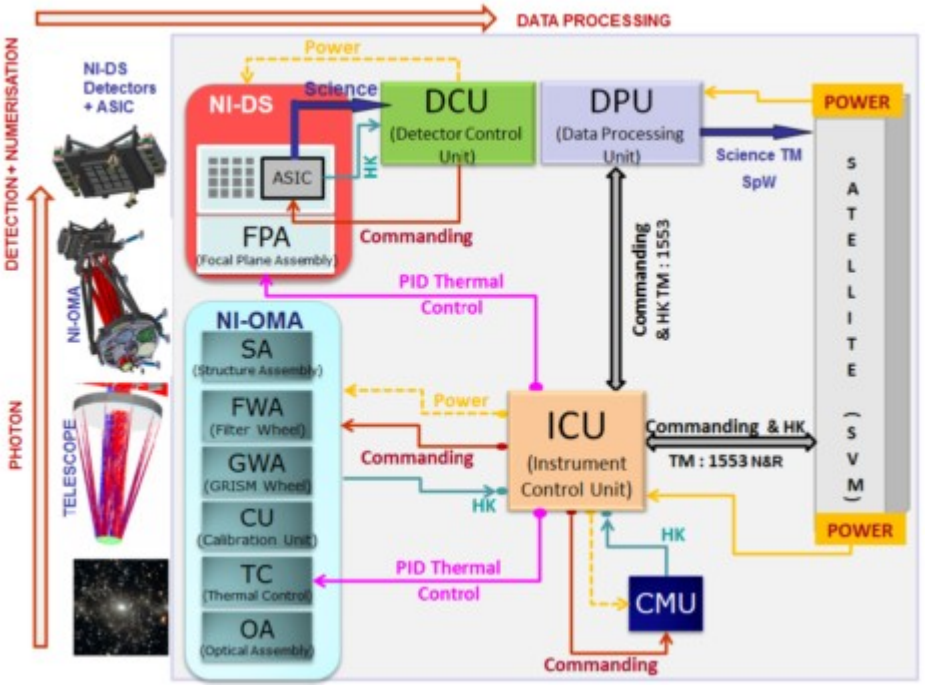
NISP throughputs through the Y, J and H filters

EUCLID NISP
Caractérisation au LAM

Instrument NISP composé de 16 SCS

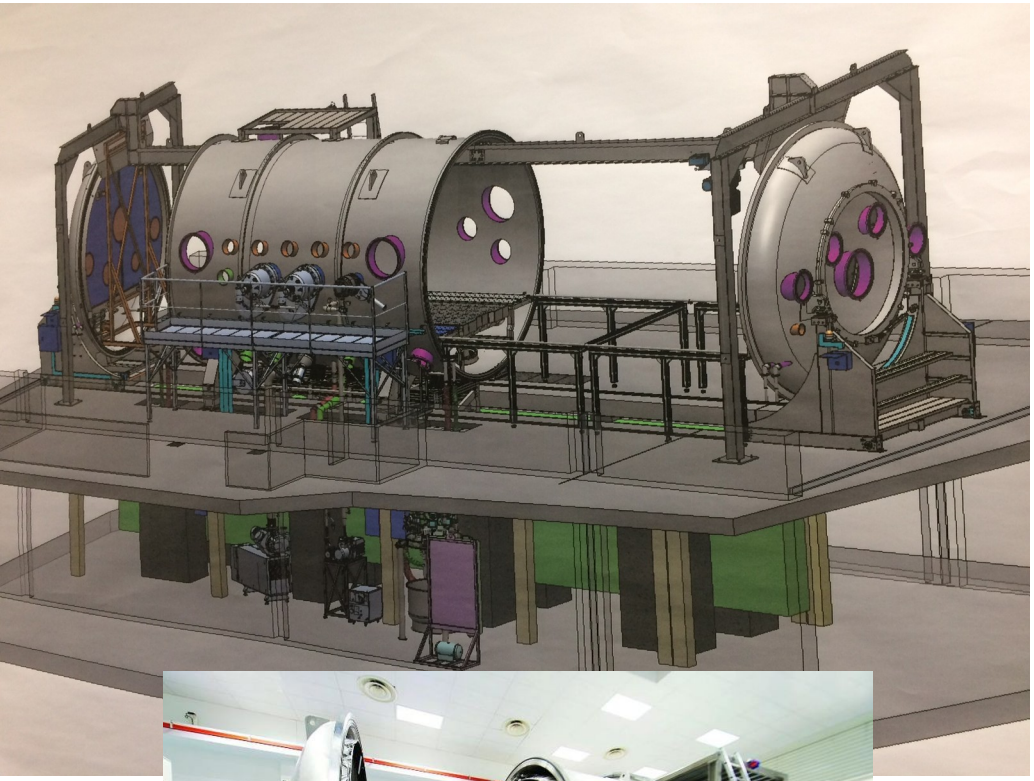


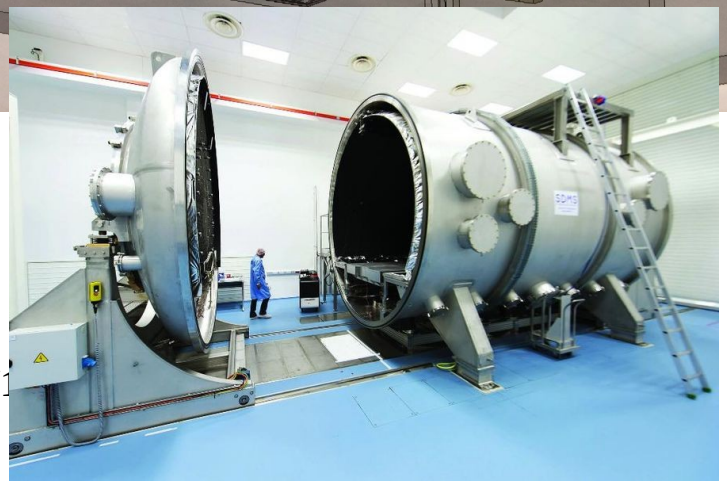




EUCLID NISP
Caractérisation au LAM

Cryostat ERIOS





Caractéristiques :

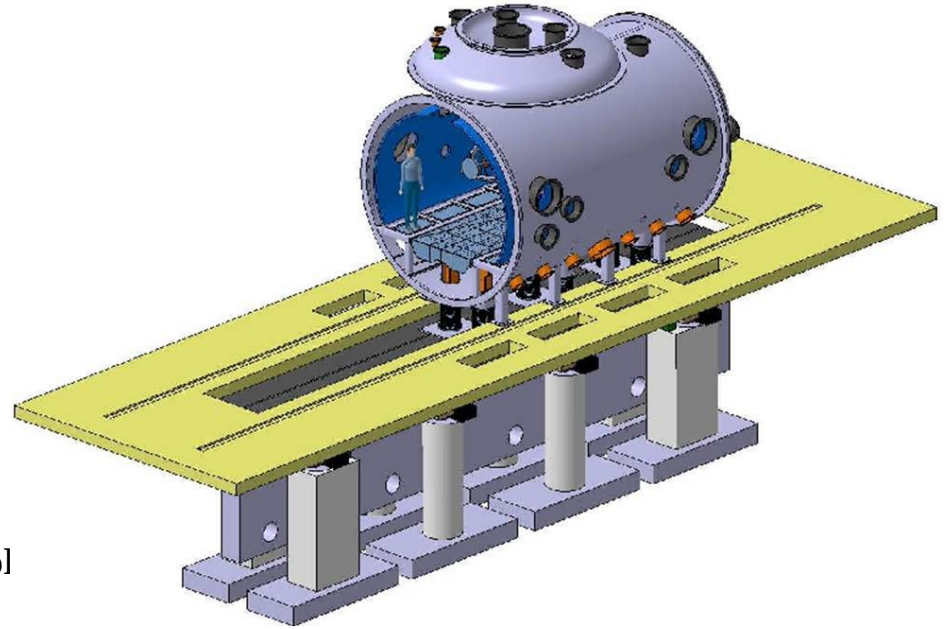
Gamme de température : 77K - 323K

Volume : environ 45 m3 utiles (longueur 6m et diamètre 2m)

Vide : jusqu'à 10-6 mbar

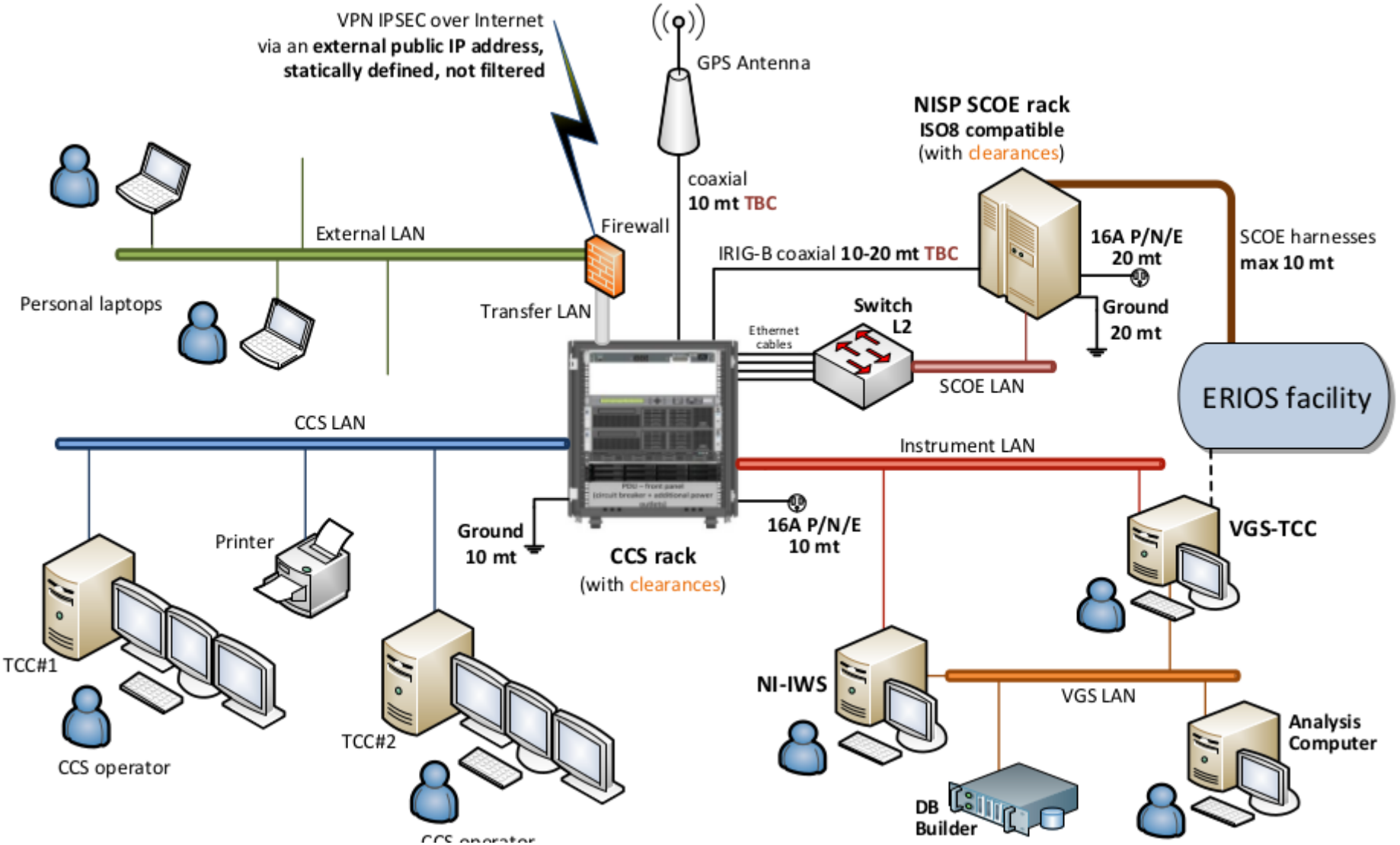
Propreté : ISO 8 - possibilité ISO5

Vibrations : inférieures à 10-7g entre 5 et 100Hz sur la table optique
Pour garantir la stabilité de l'instrument, et donc la précision des mesures, NISP sera installé sur la table reliée à une masse de béton de 100 tonnes en sous-sol, posée sur des piliers par l'intermédiaire de boîtes à ressorts



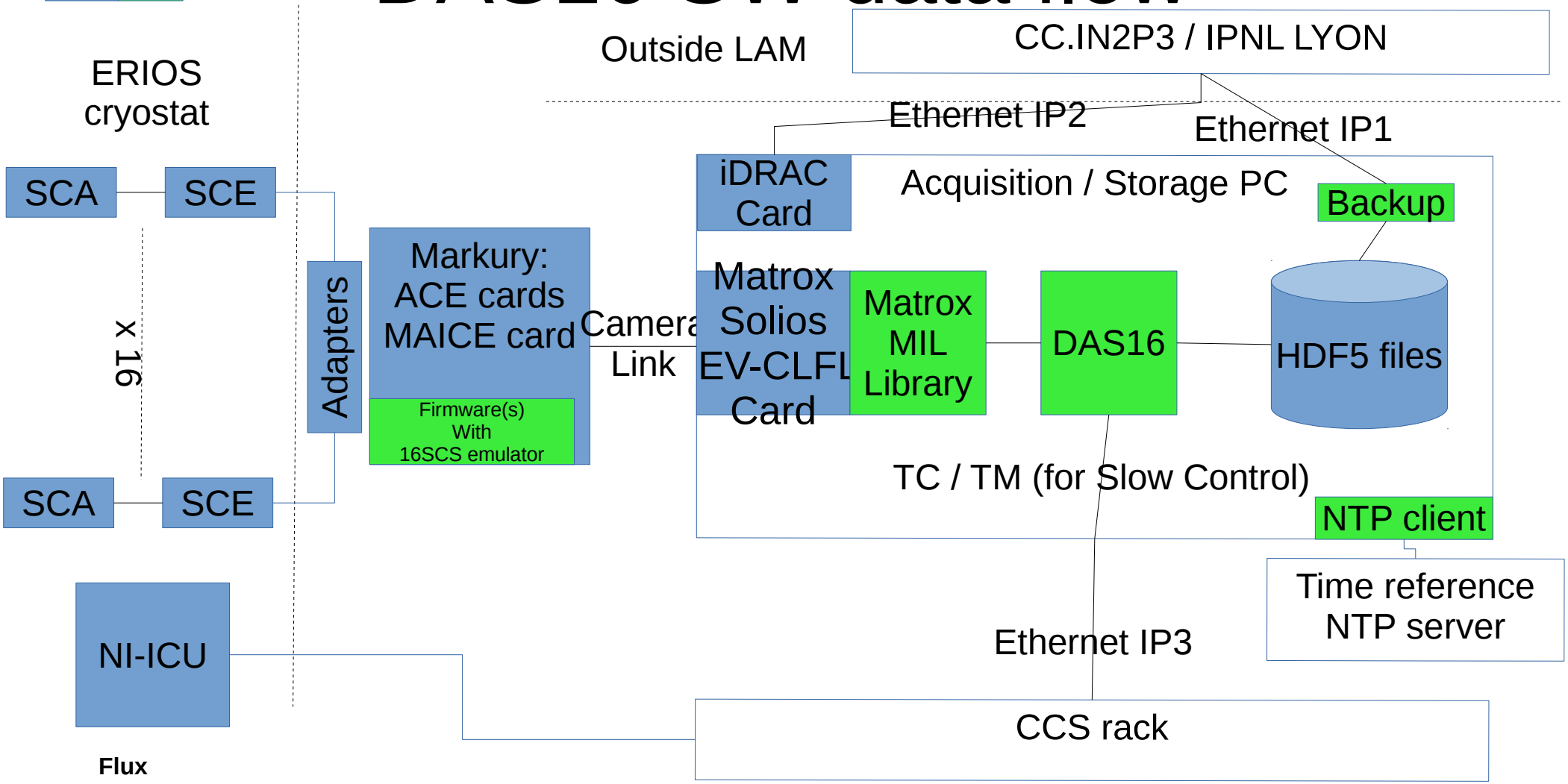
Sylvain Ferriol

EUCLID NISP
Caractérisation au LAM



EUCLID LAM 16SCS DAS16 SW data flow

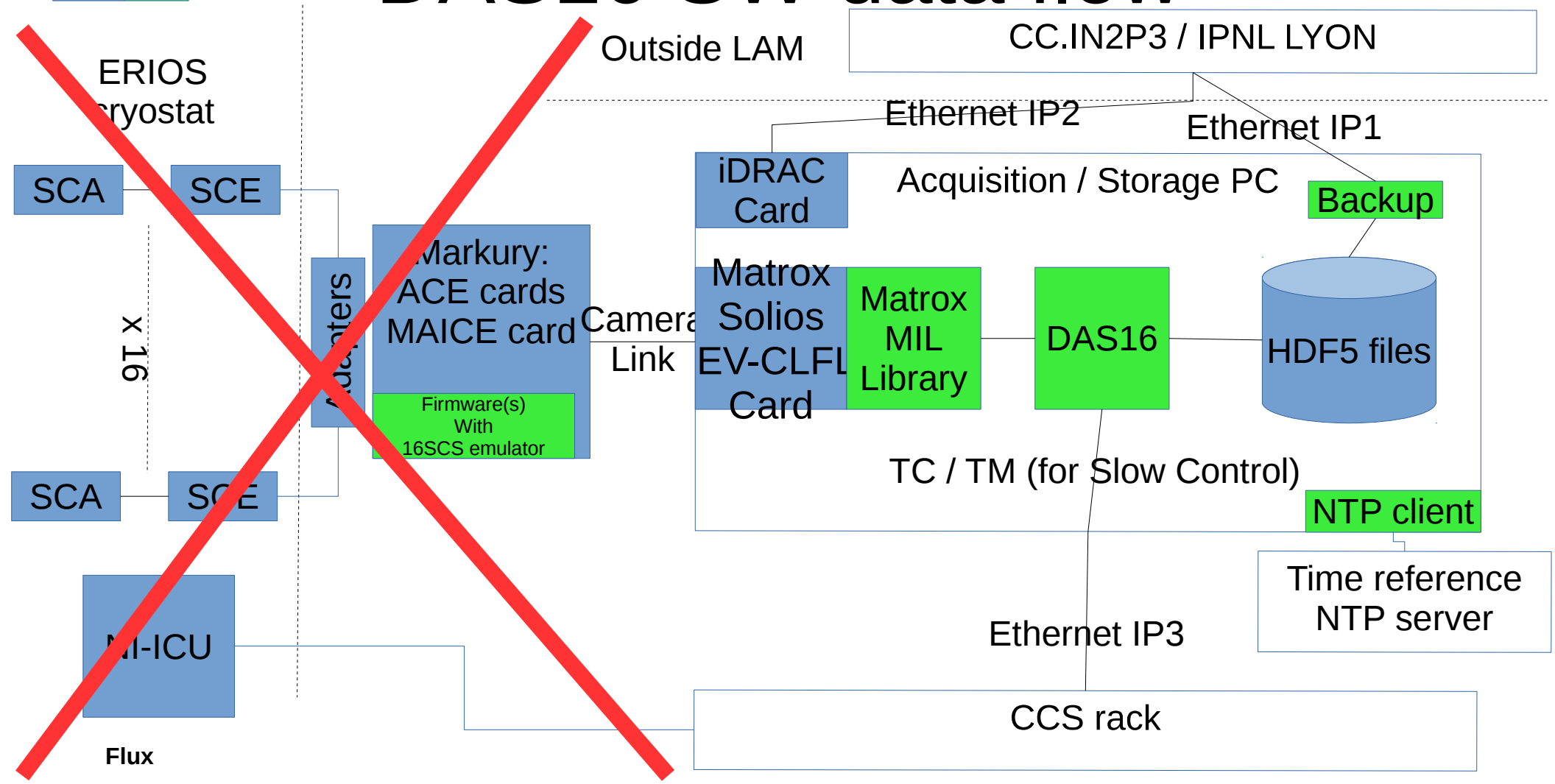
HW SW



Flux

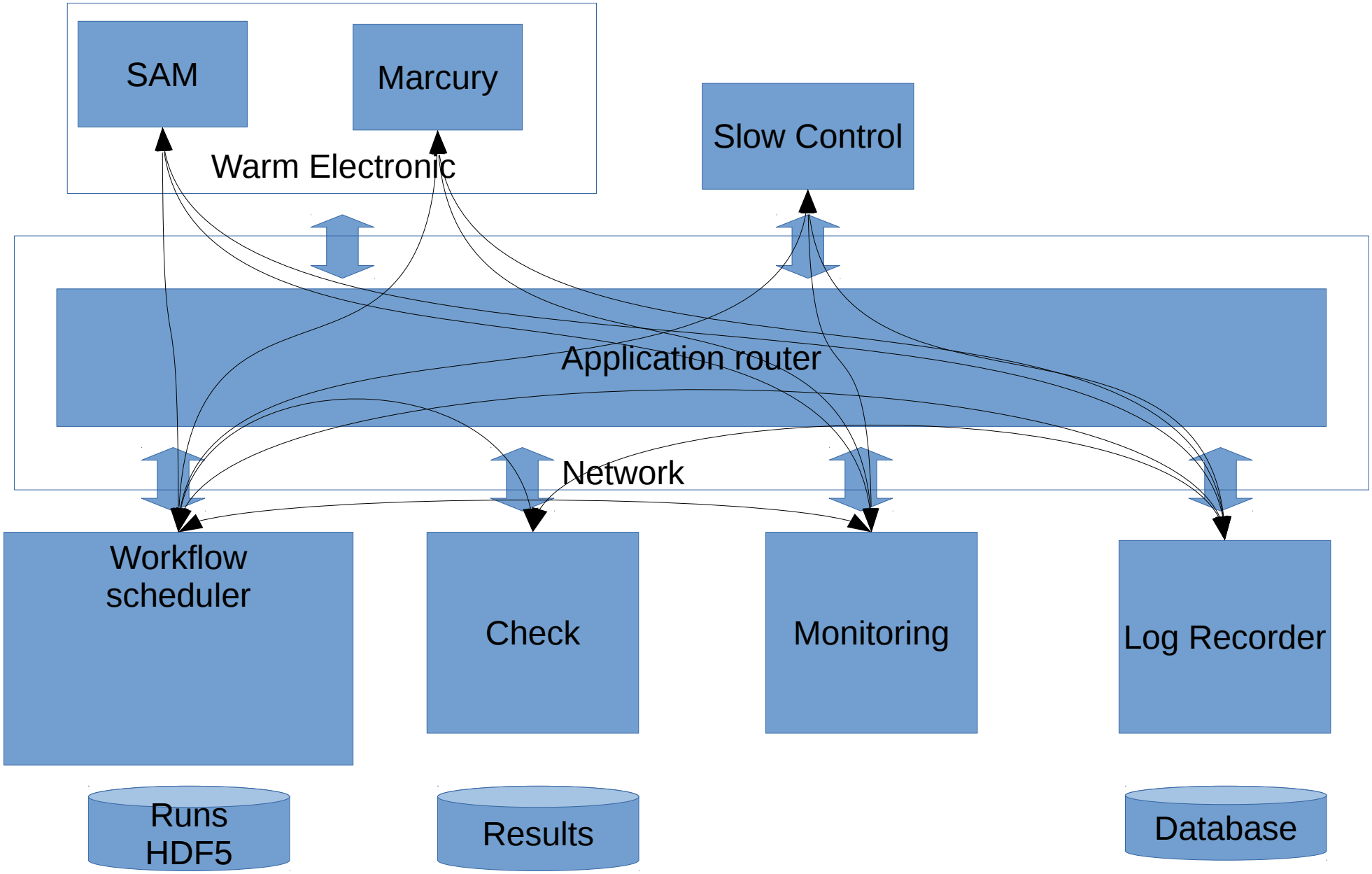
n_pixels / scs / frame = 2121 * 2048 *(with 2121= 8 header + 64 ref.channel + 2048 pixels + 1 trailer)*
n_bytes / scs / frame = 2121 * 2048 * 2 *(16bits/pixel)*
*n_bytes / 16scs / second = 2121 * 2048 * 2 * 16 / 1.4138 =* ***98,317,907*** **bytes / s**

EUCLID LAM 16SCS DAS16 SW data flow

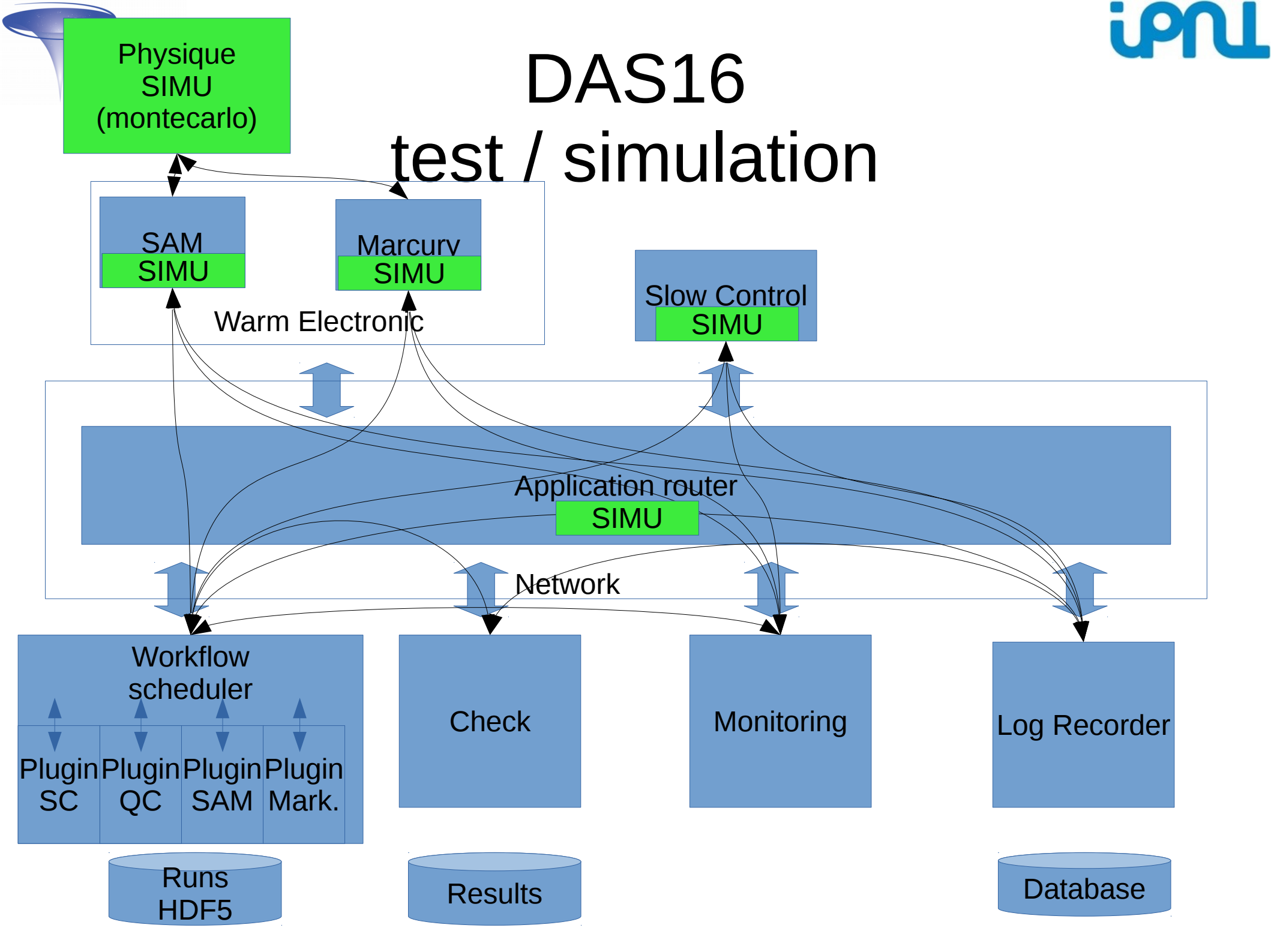


n_pixels / scs / frame = 2121 * 2048 *(with 2121= 8 header + 64 ref.channel + 2048 pixels + 1 trailer)*
n_bytes / scs / frame = 2121 * 2048 * 2 *(16bits/pixel)*
*n_bytes / 16scs / second = 2121 * 2048 * 2 * 16 / 1.4138 =* ***98,317,907* bytes / s**

DAS16


SAM
Marcury
Warm Electronic
Slow Control
Application router
Network
Workflow scheduler
Check
Monitoring
Log Recorder
Runs HDF5
Results
Database

DAS16
test / simulation



Projet EUCLID: Tests logiciels

Test et simulation sont corrélés :

- On simule des paramètres pour un test unitaire
- On simule un bruit par ex. pour des tests fonctionnels

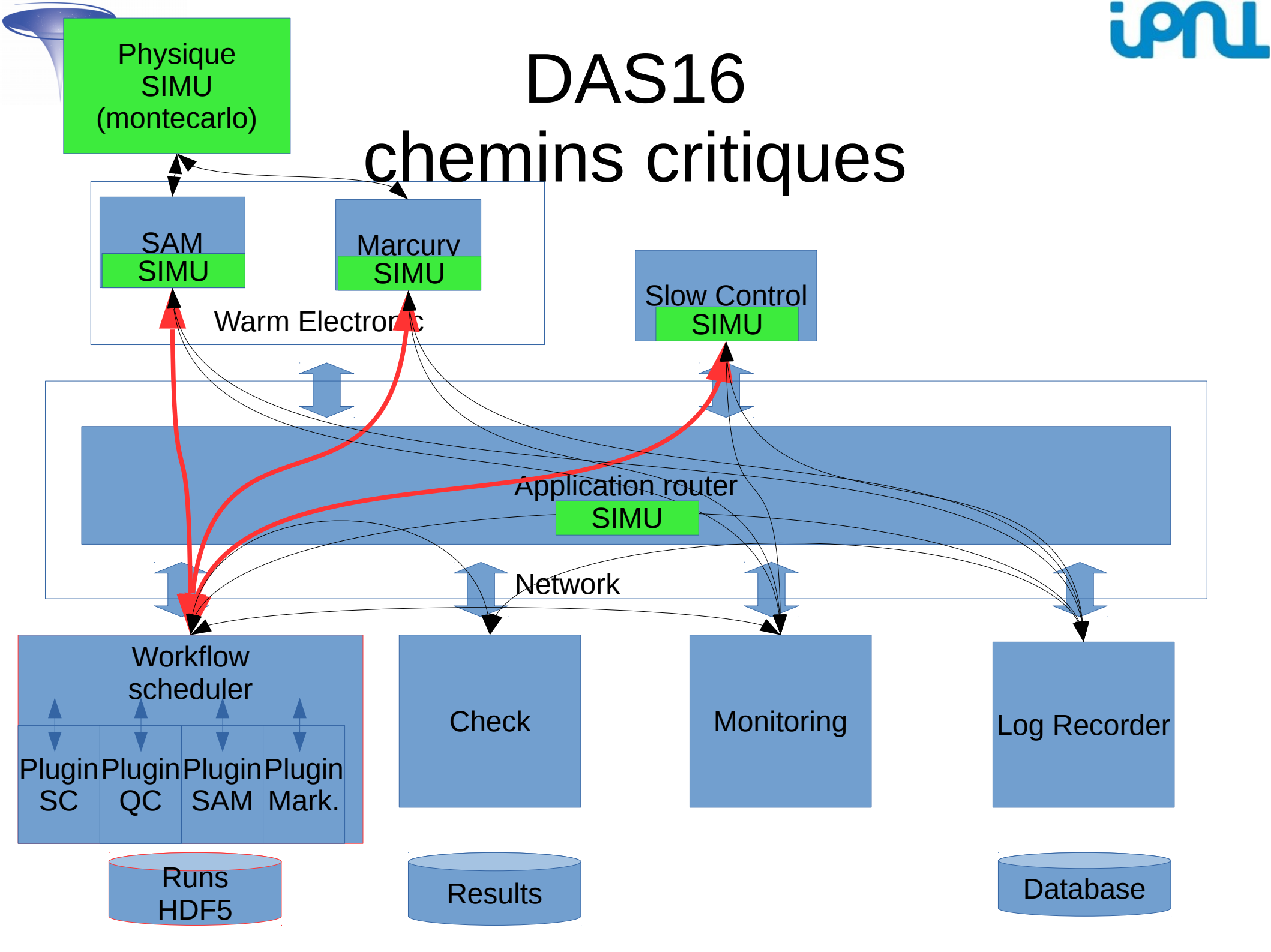
Les bouchons sont utiles quand on n'a pas accès au hardware, ou qu'une partie.

Il faut essayer de créer différents niveaux de bouchon pour pouvoir tester une ou plusieurs parties du logiciel.

Pour valider le temps-réel, il faut avoir des bouchons hardware (Tests pattern).

Mettre en place un environnement de test le plus proche de la réalité de fonctionnement du logiciel.

DAS16
chemins critiques



Projet EUCLID: Tests logiciels

Les tests sont chronophages d'où il faut définir des priorités dans les tests.

Définir les parties critiques du logiciel qui doivent impérativement être testées.

Il faut aussi les « isoler » pour qu'on ne les retouche le moins possible et éviter un risque de régression.

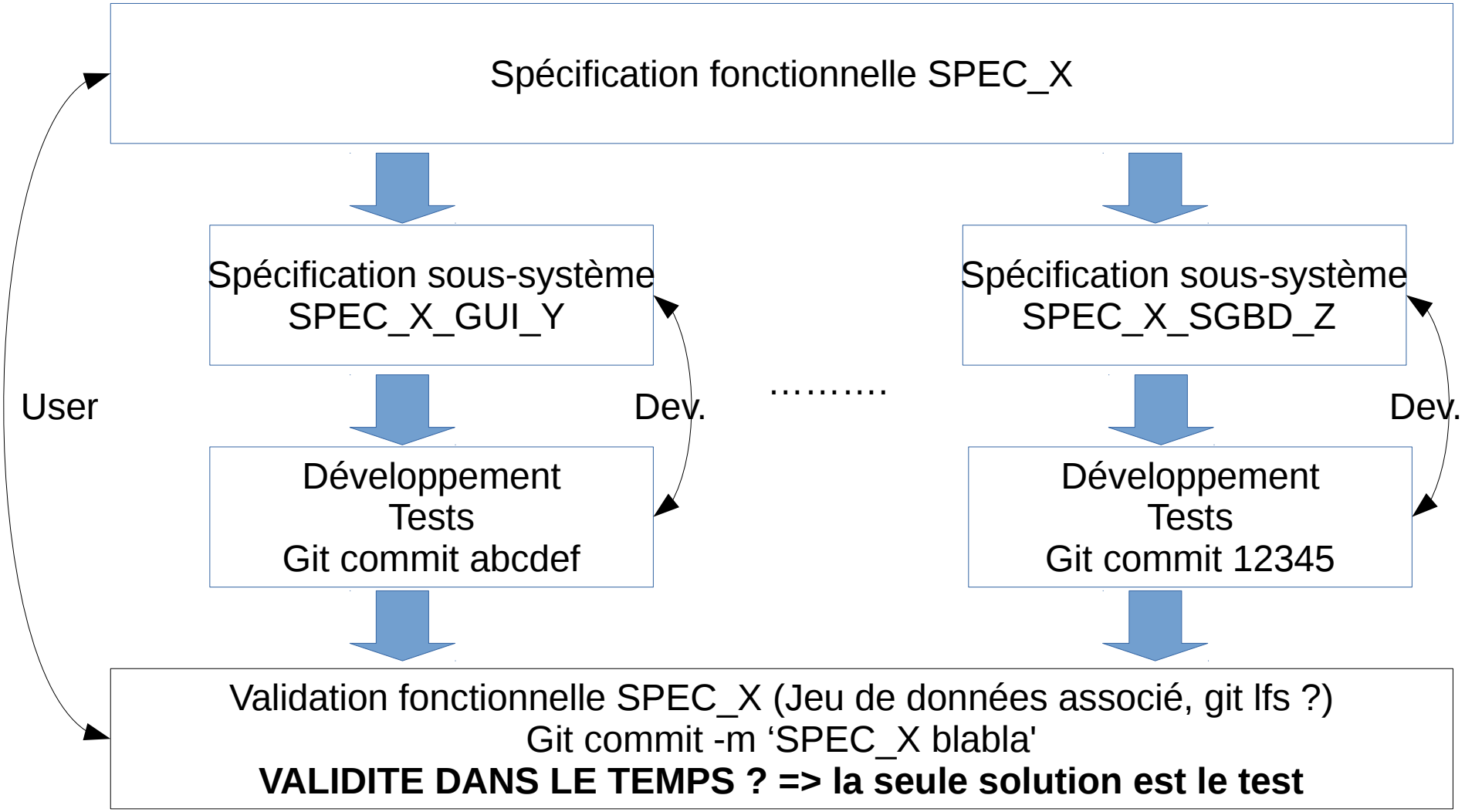
La modularité est cruciale dans la qualité et la simplicité des tests.

En acquisition, il faut préparer le travail à faire (compilation en fonctions basiques, allocation mémoire) le plus en amont possible pour pouvoir valider les étapes d'exécution de manière statique.

Pour minimiser les risques liés au temps-réel (mémoire, dépassement de temps,…), on prépare tout à l'avance.

Il vaut mieux que ça plante dès le départ qu'après 12h d'acquisition

Projet EUCLID: Tests logiciels



Projet EUCLID: Tests logiciels

Faire une hiérarchie depuis la spec de haut niveau puis en découpage de au niveau du code (avec les tests unitaires + git commit associés), puis le git commit avec le test fonctionnel qui valide la spec.
Toute spec doit pouvoir être testée et validée

Les tests doivent valider tous les bugs trouvés

Le nombre de tests est souvent corrélé au niveau de confiance qu'on a sur le code

Attention aux librairies qu'on utilise car elles peuvent génerer un bug dans l'application, ex : faire des recopies mémoires => 'out of memory' => faire des tests sur ces librairies suivant les besoins de l'appli.

Exemple ubuntu 12.04 LTS kernel : 3.11.0-15, un bug (#1275853) sur les interfaces réseau (faible débit de 130 Mb/s au lieu de 900 Mb/s sur un lien 1Gb), corrigé dans le 3.11.0-17
=> Définir le même environement de prod et de test

Comment tester avec des drivers qui imposent des versions de noyau, et qu'on ne peut pas avoir dans un conteneur

Projet EUCLID: Tests logiciels

Exemple de tests automatiques :

```
$ python tests_unit.py
----------------------------------------------------------------------
Ran 129 tests in 103.705s
OK
```

```
$ python tests_simu.py
aces [1, 2, 3, 4] asics [1, 2, 3, 4]
joined
----------------------------------------------------------------------
Ran 8 tests in 23.460s
PASSED (successes=8)
```

Projet EUCLID: Tests logiciels

Merci