ROC project development plan

X.Bonnin

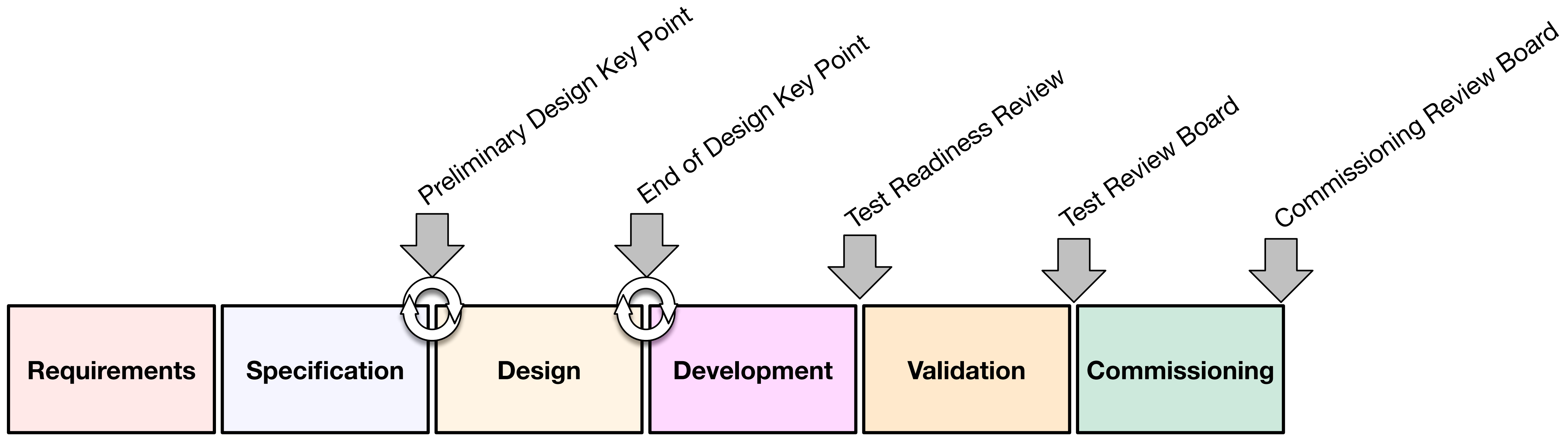
Outlines

1. ROC project management plan
2. ROC software development plan
3. ROC software validation plan (cf. S.Lion)
4. ROC operations management plan

Outlines

1. ROC project management plan
2. ROC software development plan
3. ROC software validation plan (cf. S.Lion)
4. ROC operations management plan

Software development life-cycle



- Iterations between specification/design/development due to change/unknown in the specification requirements at mission level (e.g., SBM interface, DDS client-side performance, etc.)

- Planning of "internal" validation campaign to ensure that the critical functionalities are ready for the in-flight operations.

RSS releases schedule

- Since now, 3 main releases of the ROC Software System (RSS) are planned along the development:
  - **Release RSS3**: Version "E2E" for SOV2 - 4 months (15/12/17?)
  - **Release RSS4**: Version "Ready for flight" for L - 6 months (28/09/18) — internal validation campaign
  - **Release RSS5**: "Fully operational" for L + 3 mois (28/06/18)
- (RSS main software - RODP[RCS], MUSIC, ROC-SGSE, LLVM)

RSS releases specification

- Each release shall support the minimal critical functionalities to achieve the objectives of the related milestones (summarized into the [ROC-GEN-MGT-PLN-00015-LES])

- RSS functional specification requirements are listed in the ‘ROC Software System Specification" (RSSS) [ROC-GEN-SYS-SPC-00026-LES] — labelled with unique ID

- Requirements are automatically reported and tracked into the ROC Gitlab issue tracker tool. Labelled as "spec" issues with priority level ("low", "medium", "high" et "critical") w.r.t the criticality of the requirements for a given release.

ROC development strategy

- Development timeline planning is split into 2 weeks "sprints" (tailored adaptation of the "Agile Scrum" method), with a sprint review meeting at the end (Friday at 10am)
  - Review of the tasks achievement for the ended sprint
  - Discussions about possible encountered difficulties
  - Define new tasks for the next sprint (or tasks to be continued)
  - Change/refine the development priorities depending of the constraints (project planning and personnel availability)

ROC sprint management

- Sprint tasks are managed with the ROC Gitlab issue tracker tool using Scrum board
- Tasks are labelled by priority ("low", "medium", "high" et "critical"), which can be changed from a sprint to another
- Or by type of tasks , "feature", "bug", etc.
- Rules and convention defined in "ROC Engineering Guidelines" [ROC-GEN-SYS-NTT-00008-LES]
- Tasks will be also reported as issues on JIRA (and visible from ROC issue dashboard on Confluence)

RPW Calibration Software (RCS) development activity

- The RCS dev. activity is managed in parallel, but in consistency with, the RSS dev. planning
  - Documentation (responsibilities, data products, interface, guidelines)
  - Monthly RCS meeting
  - Issues managed by the ROC-DATAPROD JIRA project
  - Dedicated dashboard on Confluence to follow issues and action-items

Development environment

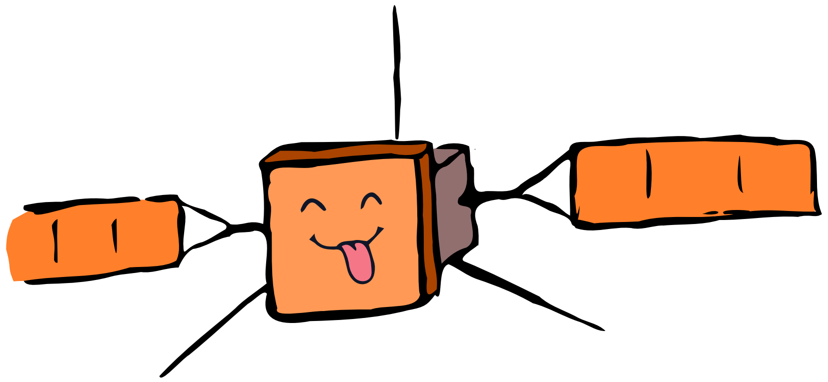
- Source codes managed with Gitlab. One Git repository per software with branching model management (see S.Lion slides)
- Continuous integration relying on Jenkins and Gitlab (see S.Lion slides)
- Dedicated development server at LESIA (also available for RCS teams)
- Dedicated database and space
- Separated software environment (virtual environment mechanism for software under Python)

Testing/pre-prod/prod. environment

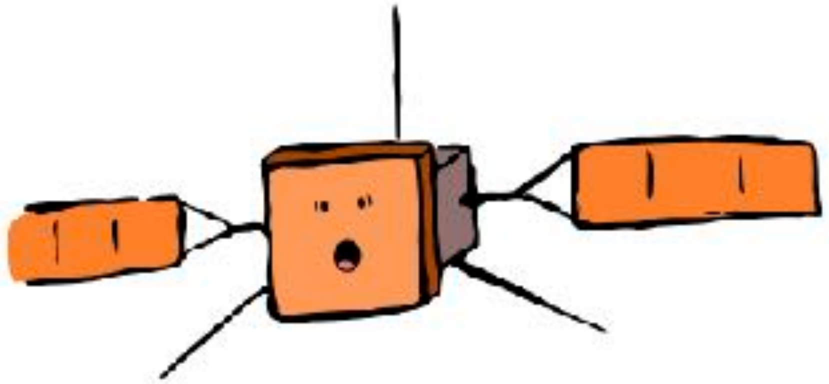
- Environment as much as possible similar
- Same server for pre-prod/prod. software
- But distinct environments (virtual environment)
- Testing/pre-prod/prod environment describing in S.Lion presentation

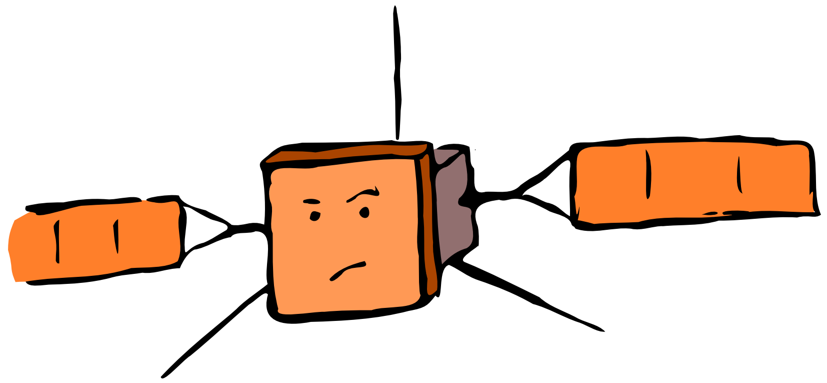
Testing/pre-prod/prod. environment

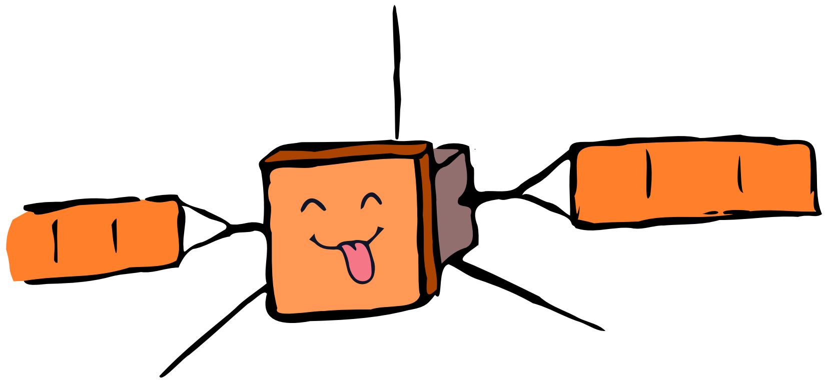
- Software installed on Virtual Machines (VM), hosted and administrated by the LESIA computer department
- Backup machines planned (laptop)
- Dedicated version of the ROC software will be deployed on Laptops (with backups) for operations at MOC (ESOC, Darmstadt, Germany) during commissioning
- MOC instance will be also tested and validated



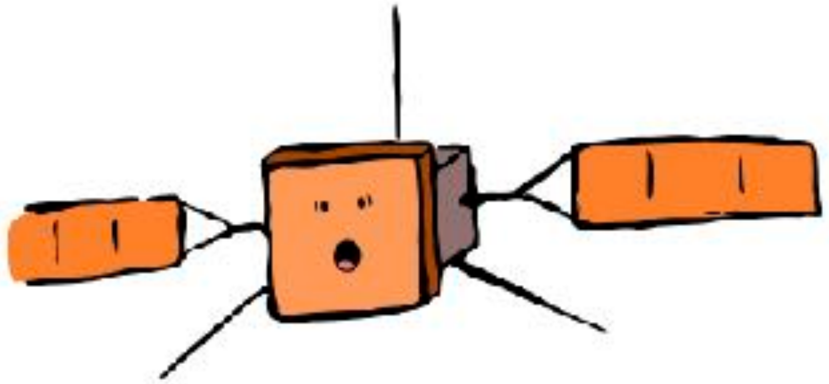
To be continued…

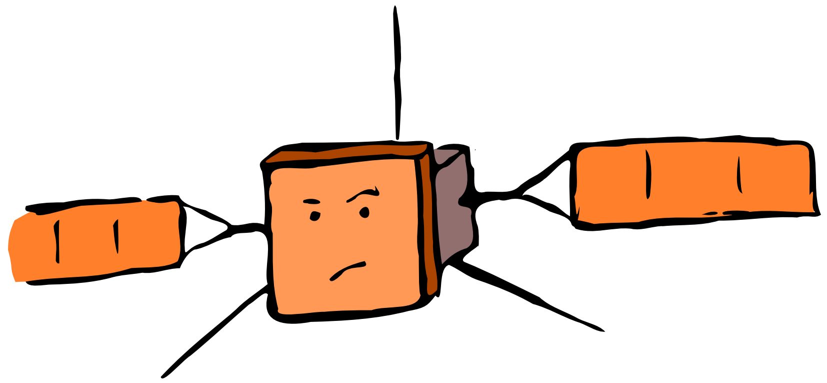






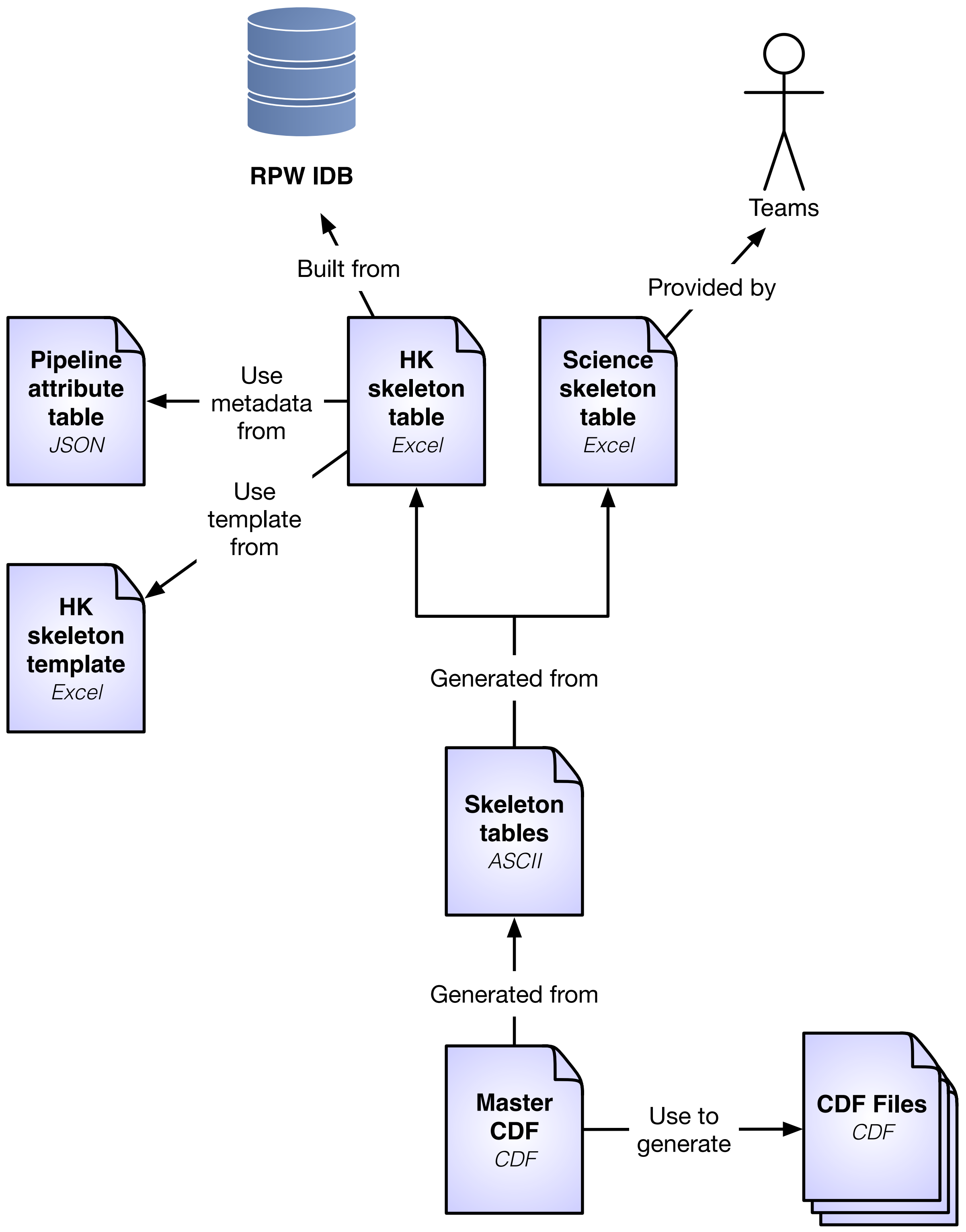
Extra slides





Production des données science: principe

- NASA Common Data Format (CDF)
- Production of CDF data files using templates (called "master binary CDF"). One master file per dataset
- Master binary CDF are generated from text file called skeleton table
- ROC (L1, HK) and lead-col (L1R, L2) teams can generate templates in Excel format
- Excel format files are converted into master CDF
- CDF templates are stored in Git repository


RPW IDB
Teams
Built from
Provided by
Pipeline attribute table JSON
Use metadata from
HK skeleton table Excel
Science skeleton table Excel
Use template from
HK skeleton template Excel
Generated from
Skeleton tables ASCII
Generated from
Master CDF CDF
Use to generate
CDF Files CDF