Tools for Collaborative Work

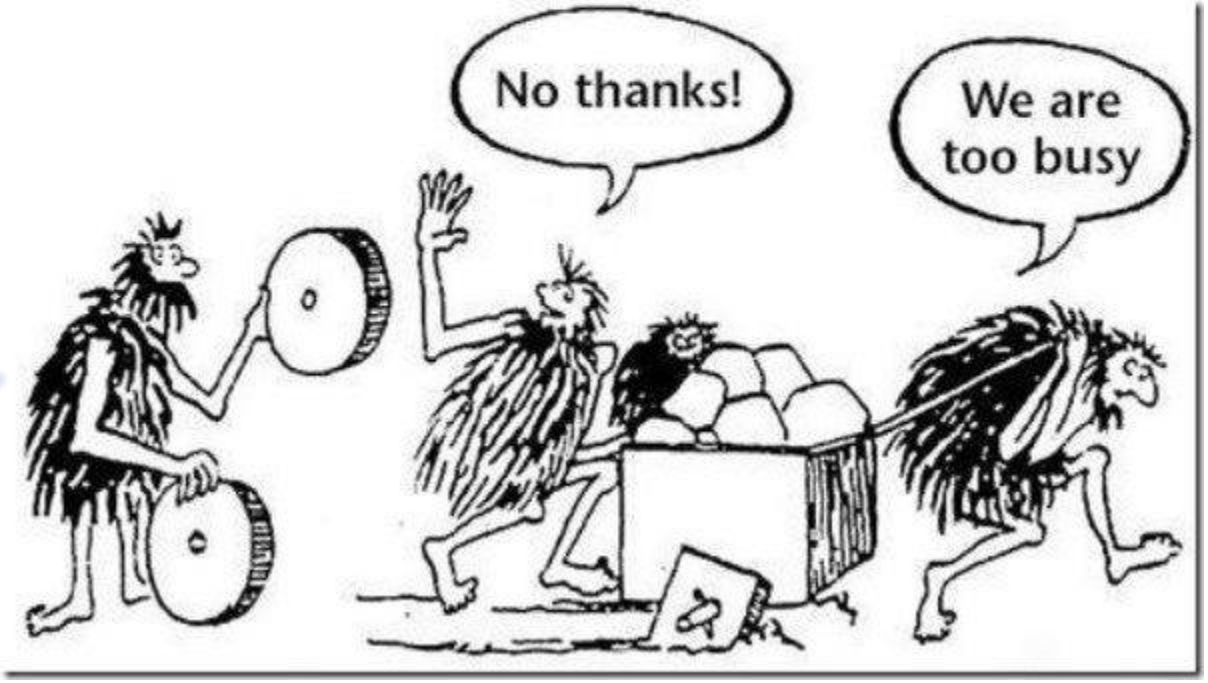
Overview and discussion

Michał Maciejewski on behalf of the STEAM
TE-MPE-PE

Motivation

- We have new colleagues in our section
- We have experience using some of the tools
- We (people in general) can be divided into two groups: *Those who **do** backups and those who **will do** backups*

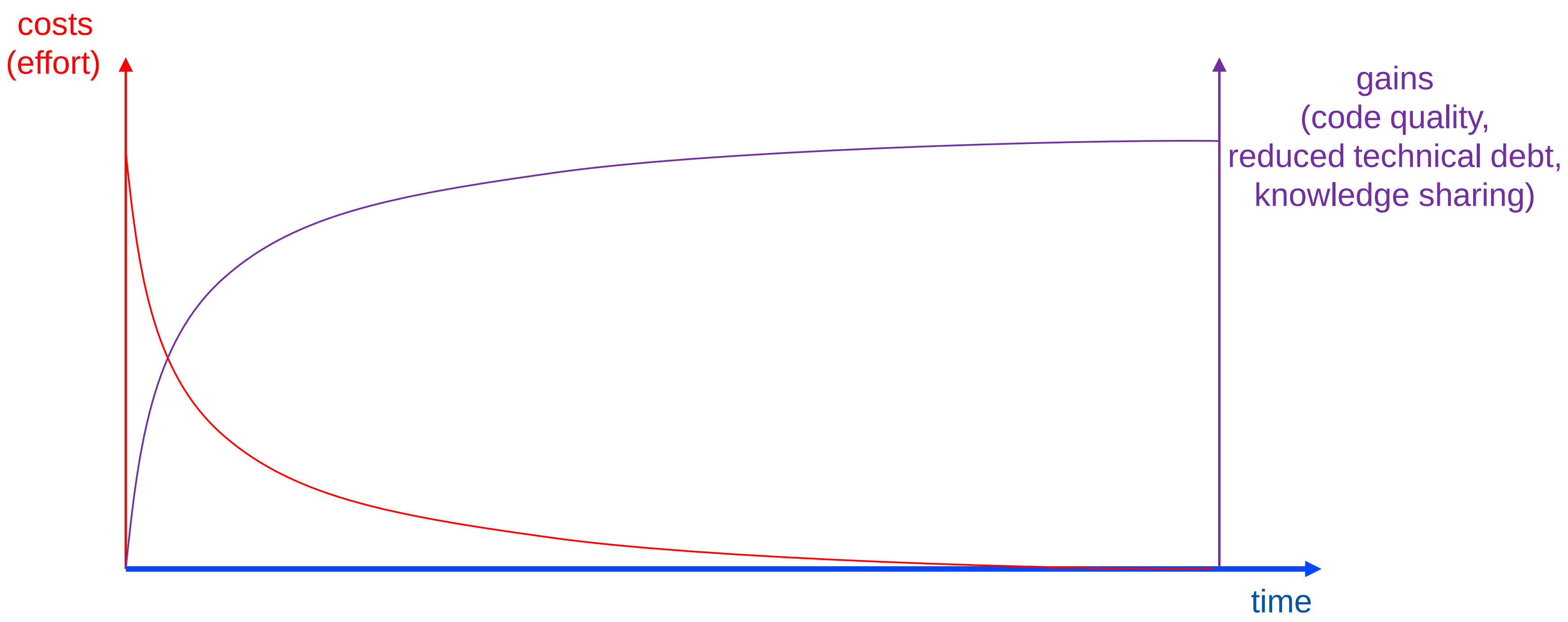
It's me
(Me how)



Source: https://www.quora.com/I-work-really-very-hard-for-CSIR-life-science-but-still-my-paper-was-not-so-good-What-should-I-have-to-do-so-I-can-crack-it

Gains vs. costs

- The concepts to be presented are based on the common sense and we've been doing some of them.
- The goal of this presentation is to structure these concepts and discuss a strategy for collaborative work.

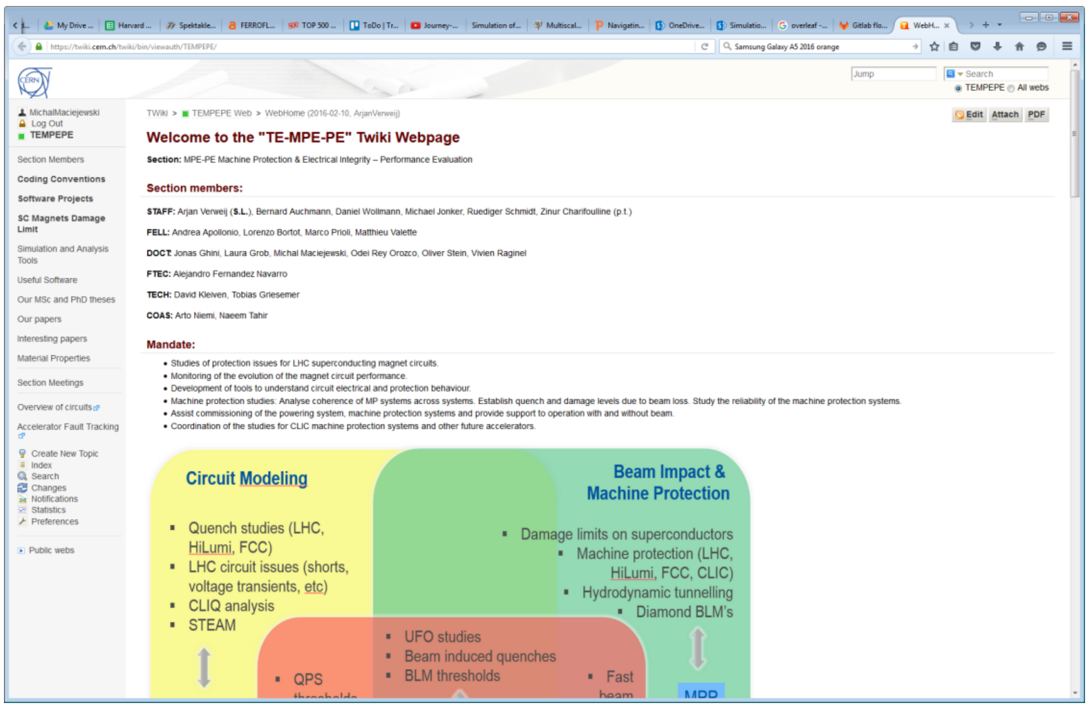


Outline

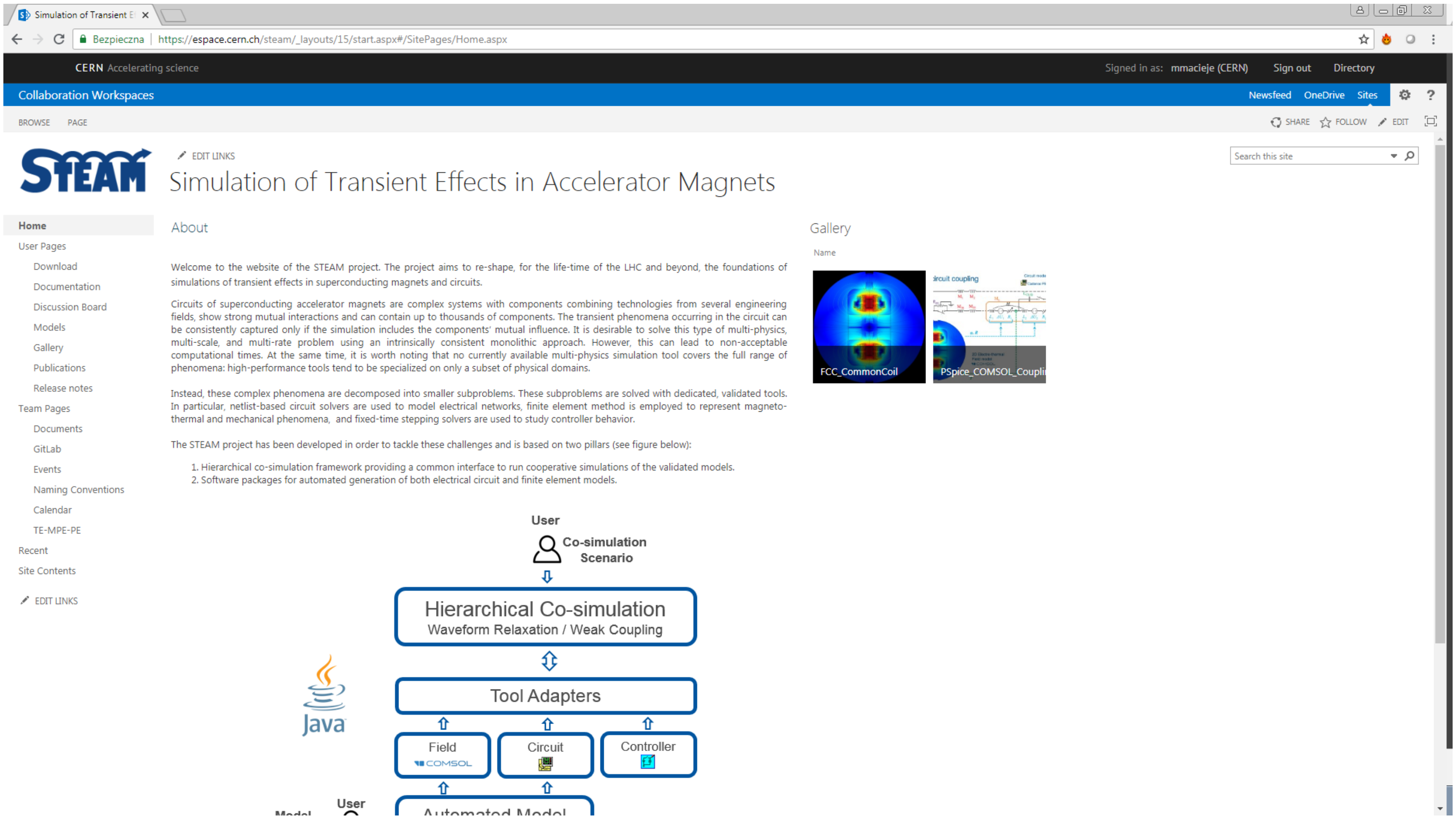
1. Twiki&Indico
2. Sharepoint
3. CernBox
4. Overleaf
5. Coding Conventions
6. GitLab
7. LHC Signal Access
8. SWAN (Service for Web based ANalysis)

Section website (cernbox.cern.ch)

- Our core product are analyses covering the accelerator needs (papers, posters, presentations, internal notes, design reports, …)
- Great platform to share our activities internally and externally!



Sharepoint* (cern.ch/steam)



*NB: CERN is in the process of Mexit

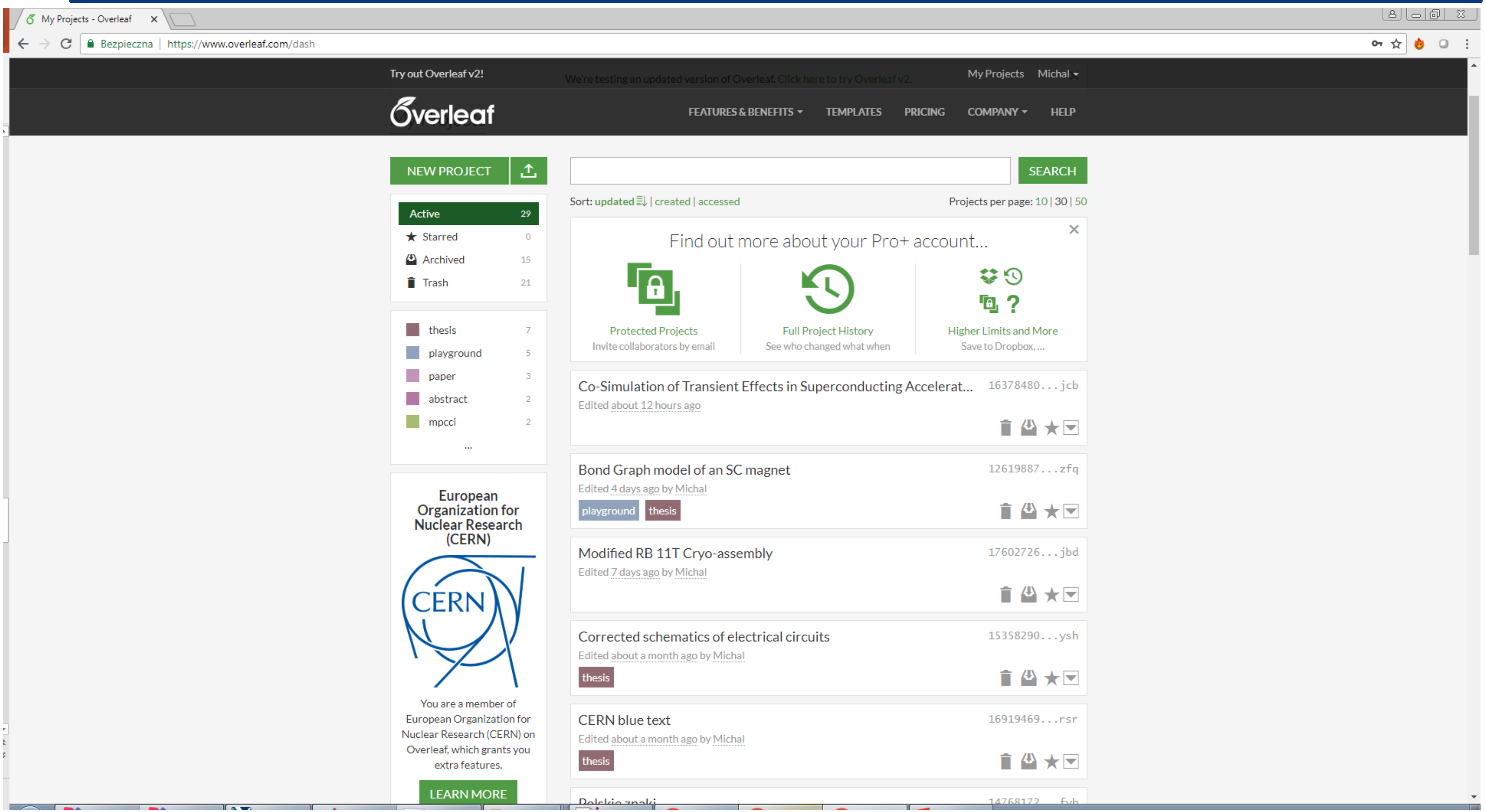
CERNBox (cernbox.cern.ch)

- Use of google drive, onedrive, dropbox, etc. is discouraged
- 1 TB per user (also personal files) / project
- Multiplatform (Android, iOS, Windows, OS, linux)
- Synchronises across multiple locations
- Stores 10 latest versions of a file

Multiple users in our section: Arjan, Bernhard, Matthieu, Michał, Christoph, Per, Zinur, Akrivi, Lorenzo, Marco, ...

Overleaf (http://overleaf.com)

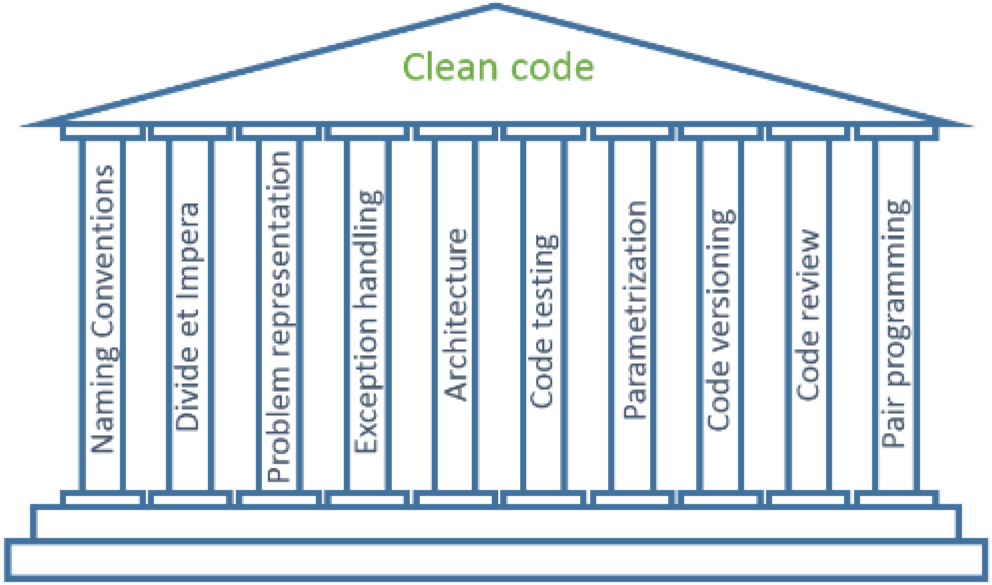
- A platform for cooperative writing of papers + paper repository!
- No more hand written corrections, multiple versions of files: *Nature_paper_MM*.pdf, *Nature_paper_MM_v1*.pdf,…



Overleaf (http://overleaf.com) + ShareLaTeX

- A platform for cooperative writing of papers + paper repository!
- No more hand written corrections, multiple versions of files: *Nature_paper_MM*.pdf, *Nature_paper_MM_v1*.pdf,…

Good practices, continuous integration workflow



GitLab

Code repository
Code review
Continuous integration

Scrum methodology
Mon-Wed-Fri stand-up meetings @10AM

Best Practices for Scientific Computing

http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1001745

Box 1. Summary of Best Practices

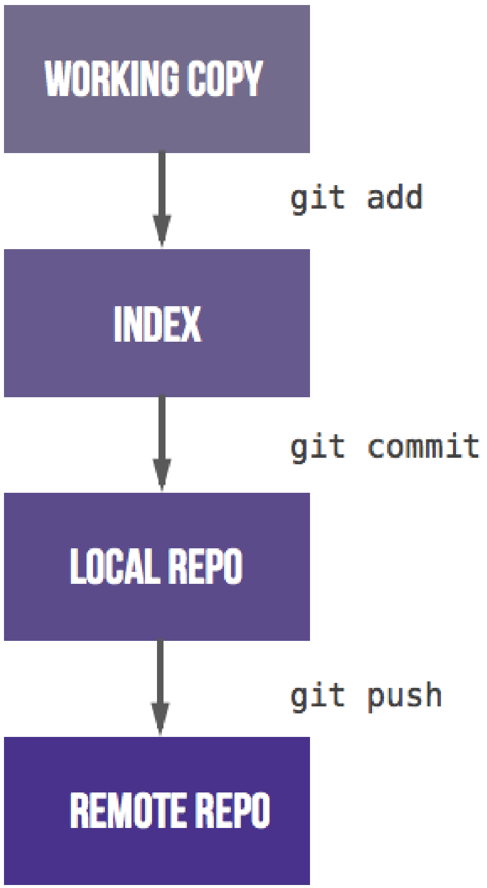
1. Write programs for people, not computers.
    a. A program should not require its readers to hold more than a handful of facts in memory at once.
    b. Make names consistent, distinctive, and meaningful.
    c. Make code style and formatting consistent.
2. Let the computer do the work.
    a. Make the computer repeat tasks.
    b. Save recent commands in a file for re-use.
    c. Use a build tool to automate workflows.
3. Make incremental changes.
    a. Work in small steps with frequent feedback and course correction.
    b. Use a version control system.
    c. Put everything that has been created manually in version control.
4. Don't repeat yourself (or others).
    a. Every piece of data must have a single authoritative representation in the system.
    b. Modularize code rather than copying and pasting.
    c. Re-use code instead of rewriting it.
5. Plan for mistakes.
    a. Add assertions to programs to check their operation.
    b. Use an off-the-shelf unit testing library.
    c. Turn bugs into test cases.
    d. Use a symbolic debugger.
6. Optimize software only after it works correctly.
    a. Use a profiler to identify bottlenecks.
    b. Write code in the highest-level language possible.
7. Document design and purpose, not mechanics.
    a. Document interfaces and reasons, not implementations.
    b. Refactor code in preference to explaining how it works.
    c. Embed the documentation for a piece of software in that software.
8. Collaborate.
    a. Use pre-merge code reviews.

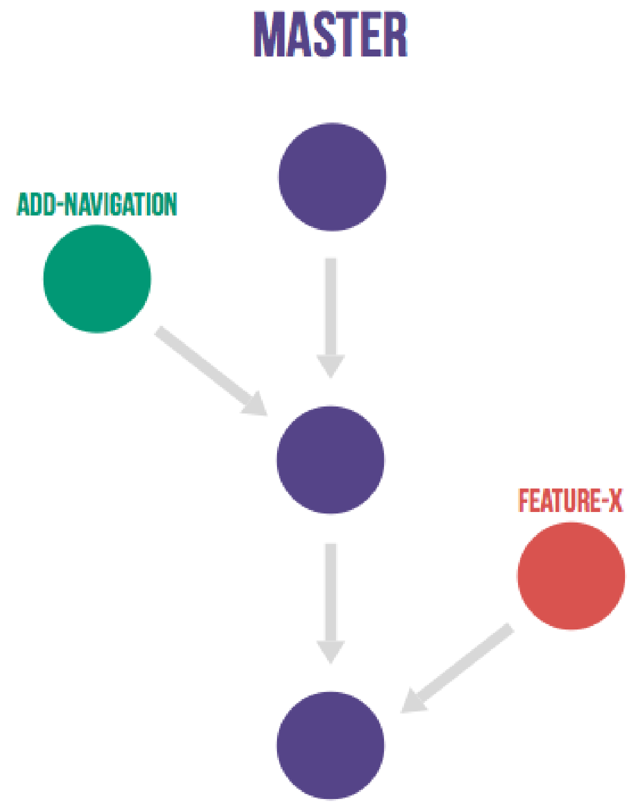
Citation: Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, Guy RT, et al. (2014) Best Practices for Scientific Computing. **PLoS Biology** 12(1): e1001745. https://doi.org/10.1371/journal.pbio.1001745

Simplicity is the art of maximizing the work not done

- KISS – Keep It Simple, Stupid!
- Recognition and use of design patterns
- Code review with experts (TE-MPE-MS, EN-ACE-EDM)
- Search for canonical, math-based problem representation
- Internal code refactoring
- Static code analysis with **sonar qube** (code duplications, code smells, complexity)
- **Humility in programming (complicated solution is not impressive...)**
  - http://labviewjournal.com/2013/05/humility-1/
  - http://labviewjournal.com/codereviews/Code%20Review%20Presentation.pdf
  - https://www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD340.html

Gitlab (gitlab.cern.ch) – versioning!





Several users in our section: Matthieu, Martyna*, Philippe*, Michał, Christoph, Per, Zinur, Akrivi, Laura, Lorenzo, Marco, Jonas, Alejandro*, ...

https://twiki.cern.ch/twiki/bin/view/TEMPEPE/CodingConvention

LHC Signal Access

- In our section we are often times querying signals from the LHC (FPA, QPS, FGC, BLM and many more cryptic abbreviations)
- There is a number of custom tools to access Post Mortem (Java, LabVIEW) or CALS (TIMBER)
- Within our group (MPE-MS) there is a PM REST API
- BE-CO develops NXCALS
- We are creating a light-weight API to homogenize signals

http://pm-api-pro/v2/

http://pm-api-pro/v2/
pmdata/signal?system=FGC&
className=51_self_pmd&
source=RPTE.UA27.RB.A23
&timestampInNanos=1529792241640000000
&signal=STATUS.I_MEAS

Several users in our section: Matthieu, Philippe, Michał, Christoph, Per, Zinur, Akrivi

SWAN (http://swan.cern.ch)

- Analyse data without the need to install any software
- Access experiments' and user data in the CERN cloud
- Share with colleagues
- Notebook = code + output (in one file!)

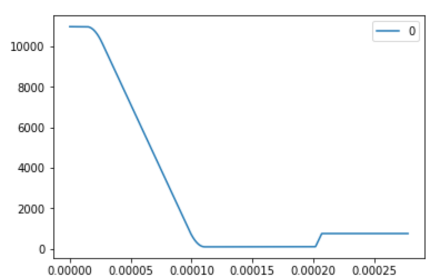
Retrieve and plot DFLAS.7L2.RB.A12.LD1:U_RES in sector 12 at 2018-07-01 10:00 - 11:00

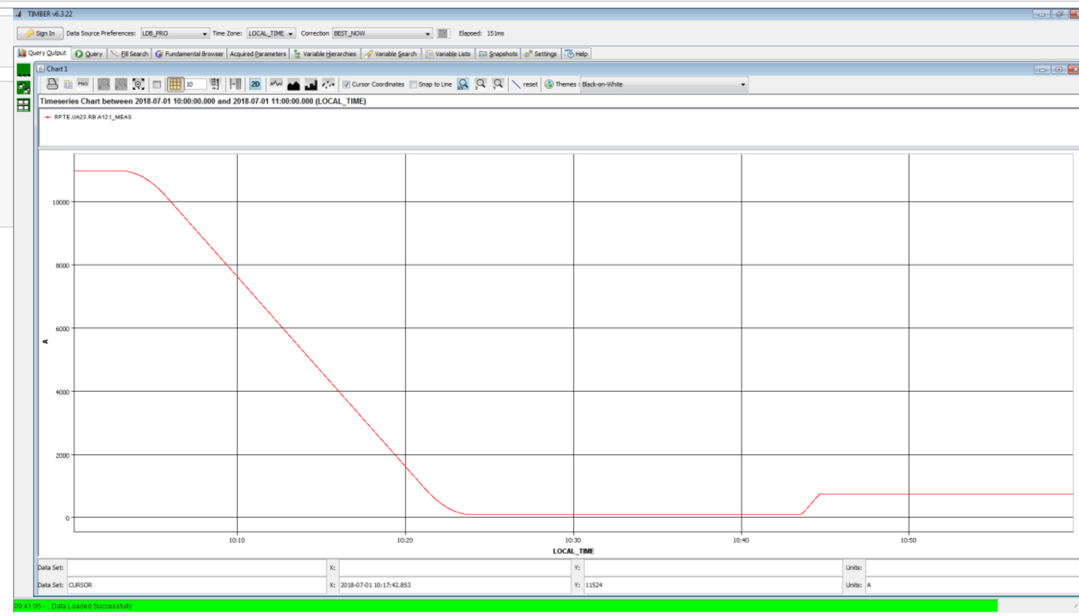
```
In [22]: # RPTE.UA23.RB.A12:I_MEAS
         t1 = '2018-07-01 10:00:00.000'
         t2 = '2018-07-01 11:00:00.000'
         d = ldb.get('RPTE.UA23.RB.A12:I_MEAS', t1, t2)
```

```
In [28]: time = d.get('RPTE.UA23.RB.A12:I_MEAS')[0]
         current = d.get('RPTE.UA23.RB.A12:I_MEAS')[1]
```

```
In [29]: time[:] = [t - time[0] for t in time]
         # Convert from s to h
         time[:] = [t/3600 for t in time]
         # Create a data frame
         df = pd.DataFrame(current, time)
         # plot
         df.plot()

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7ffa9d072a58>
```





Several users in our section: Matthieu, Michał, Christoph, Per, Zinur, Akrivi

Be a T-shaped person