Work Mechanics

Likes

- The interaction between the team members is easy (direct contact, email, mattermost, etc...)
- The tasks and responsibility of each team member are well defined
- Criticism well received and in general productive
- Everyone in the group can speak their mind and even newcomers' opinions are listened to
- People always available for a talk or help (everyone is friendly and helpful)
- Constant feedback about our work among the members of the team
- Freedom for decision making in each work area
- For external contributors, regular ROOT meetings is the only chance to participate in live team discussions

Likes

- Reviewing old Jira issues every week is a good idea
- Not much formal overhead
- Decisions are justified and reasons communicated
- Relaxed working environment
- Opinion of everyone has a weight, from summer student to staff

Dislikes

- JIRA notification system is terrible
- Development might be refused in PR (losing time and work)
- PR might create conflicts when several people work on the same piece of code

Suggestions

- PRs should never be opened/validated/merged all by the same person. It happens all the time and it makes it difficult to correct bad practices and it lowers the average quality of the commits and of commit messages.
  Moonshot: we could require 1 approval for PRs to be merged, even if it comes from people that do not work in that area of ROOT -- it's fine if the review is not in-depth, it's still better than no review, and this way developers might get a little familiar with parts of the codebase they never look at, or might just learn some feature of C++ or python they didn't know. Responsibility of bugs and breakages introduced would still lie with the code owners.

Suggestions

- If not dailies, we should at least have (more) regular round tables, with developers explaining things in a way that everybody understands (just a quick "what I want to achieve, why, what problems I'm encountering" rather than long, detailed technical summaries that only experts understand).
  (BB: we had daily scrums in the past)

Suggestions

- A documentation for developer could be cool for newcomers or some hints on the development
- Automatic coding conventions
- Clean the code (there are a lot of duplicate functions, or ugly code that can be changed in order to increase the maintainability)
- 1-3 days hackathons, where we lock down ourselves some place, perhaps together with a few key users and contributors, and work on a predefined set of problems
- Non-work related team outing (skiing, hiking, etc.)

Improvements

- PR Review (Code owners are cool, but sometime it allow ugly of unfinished things to be merge, by the code owners themselves and this often result in things broken or not working as intended)
- Roadmap (it would be cool to see that only this many tasks remains before the release, having a progress bar seeing that it remains only 3 features, would be cool (Like GitHub milestones)
- Kanban (workflow management method) it would be nice to see where everybody is at and what's remains to do, for example it would allow that 2 people are not working on the same files at the same time to avoid conflicts
- Improve the communication in the whole team when changes that could affect every area are about to kick in

Improvements

- More personal involvement in the ROOT forum, Github PRs and CI (don't wait for the shifter to send messages...)
- PR are merging slowly: For ROOT to be more competitive, it is important to be more dynamic and accept quicker contributions. Maybe having more reviewers and/or putting milestones for review could help to converge quicker, or maybe shifters can ping reviewers.
- Better involvement of experiments: It is important that we can have discussions with other representatives from experiments or even have a topical meetings (once per month) on which we can invite people who are interested in a particular area.
  (BB: "ROOT Experiments meeting"?)

Improvements

- Many of the team members are "siloed" in a specific area - people are often unaware of the current project objective, progress, how design choices are made/technical issues are solved etc. faced by other team members. While it is impossible to know everything, we could improve the communication of high-level information in these aspects

Improvements

- A detailed plan of work for every member of the team is missing. We know we have two releases per year so every member should show up in the PoW meeting at the beginning of the year with a concrete plan of what will be included in each release for their area, together with work milestones that are not trivial, neither too ambitious, and that can be easily measured. After each release, we should have a control meeting where the milestones are examined and we evaluate where we are with respect to what was planned. It looks like in the PoW we set objectives that are too vague and that we never really go through them again (where they accomplished? And if not, why not? Did something else come out that needed to be tackled? What was it?). This proposal would help/encourage/force team members better organize their time and also help the project leader better measure the team's performance.

Improvements

- Improve how we handle the situation of a person leaving the team. This year we have been hit hard by this. In such a situation, it should be discussed during a team meeting, before the person leaves, what the responsibilities of that person are (including, and very importantly, supervision) and how it makes sense that we redistribute them over the team. The point is that absorbing the blow of a teammate leaving should be a team effort, and team members should be encouraged to participate in that effort and state how they can contribute. It is not good that everything falls on the shoulders of a very reduced group of people who are already very busy, sometimes even in an implicit or taken-for-granted way. Of course, the aforementioned meeting can be prepared with prior private conversations between the project leader and members of the team, if necessary.

Improvements

- Involvement in trainings: It looks like there are a lot of opportunities to promote the new nice ROOT features, promoting them to trainers or presenting them in front of students. (It could also allow to keep up-to-date a ROOT training material).
- Deprecation procedure for the tests: ROOT test suite could be reviewed and maybe very old tests should be replaced/removed by modern ones.

Improvements

- Improve the ROOT team meeting:
  - More general discussions
  - Decision making could be improved by discussing more the relevant issues at the meetings
  - Mail/Mattermost/github discussions ok but only for lower level and technical issues
- Code reviews are crucial! We have very strong requirements regarding backward compatibility, so putting in code should be always taken very seriously. There is some uncontrolled code growth resulting in an unmaintainable part of ROOT. A solution could be enforcing a code review on GitHub level and block the merge otherwise.
- Protect master on GitHub, always. Allow only PRs.

Improvements

- Everything should go into an experimental namespace first. Goes inline with the points above.
- Have more frequently meetings in small circles for different parts of ROOT
  (BB: again - daily scrums?)
- Introduce a mechanic that enforces people to report what they have done over the last weeks/months. For example have yearly or half-yearly presentations of each member showing what they achieved and what they plan to do in the next months. It is crucial to have more transparency in the team regarding the work of each individual team member.

Improvements

- Feedback, feedback, feedback! From within the team and as well from users. We are not doing badly but we should always try to improve and push further because this makes ROOT better and let's us develop in the right direction with the limited manpower we have. That means being very proactive in having internal meetings and discussions of your own work but as well presenting our work to users ala ATLAS workshop / carpenter workshop / CMS introductory courses / ...
- There is so little overlap that there is no (friendly!) criticism / cross checks on what's going on in other fields. Looking more into what people are doing would trigger some discussions / reflections on what would be the best / most important to do next, and we would be able to focus better on the most important parts. It might also improve code quality / programming models etc

General comments

Some things improved. Still missing well-described design ideas which have proper life-cycle (root-evolution) especially for the long-term features. If we want to be able to make more drastic changes we need to know at least the experiments codebases or at least to have good relations with their maintainers.

Root-evolution + spending 4-5 hours a week improving long-lived external-to-ROOT codebases should be the way to go.

ROOT should be not considered as an "SFT" project as it never actually was. People should be open to ideas and cooperation from the outside and also give chance other entities to develop ROOT core engineering teams, because the resources in SFT are limited.

Conclusions/Discussion

- Very positive feedback overall
- Very constructive comments, suggestions and improvements
- Highlight
  - Missing clear and detailed individual plan of work?
  - More activity reports
  - Better cross-domain (cross team members) communication
  - Re-introduce daily scrums (per domain)?
  - Better handling/response time for PRs