Foreword / Disclaimer

I'll talk quite a lot today. Trying to summarize. Trying to propose improvements. Most things are based on discussions with others.

You might disagree with some aspects.

You might find something essential missing.

…

- ***Please, don't take anything personal!***
- ***Don't think I left something out intentionally.***
- ***Just speak up and say what you think!***

Development coordination

M Hoelzl

Why is this so important?

- **Development is more active than ever**
  with various small and large pull requests in parallel

- **We need to avoid having pull requests open “forever”**

- **We need to avoid solutions which (seem to) work, but…**
  - do not work together with other developments
  - cause problems for other applications
  - narrow down the code flexibility/generality
  - actually were broken from the start

Main measures

- **Discussions via Jira before starting to implement** (directly mention someone via @name)
  - I propose we insist on that from now. No pull request without jira.
  - Please even report issues, if you don’t know how to fix them such that someone else can take care of it.

- Automatic **regression tests** (cover only some code applications)

- Code **reviewing**

- **Development meetings**

Where are we standing and where to go?

- **We are doing an increasingly good job in my eyes already, but there's plenty of room for improvements**

I provide examples in the following for problems we see regularly
(I have made quite a couple of them myself. My intention is absolutely not to blame anyone, just to learn from our past experience.)
… and propose how we might try to avoid them in the future.

(This is not a complete list of problems, but includes the most important issues, I think)

Insufficient description about what a pull request is supposed to do

- **Makes reviewing unnecessarily hard**
- **Missing records about the changes**

- **Make sure to explain well <u>why</u> this is needed, <u>what</u> is done, and <u>how</u> you do it.** Explain not for yourself, but in a concise and comprehensible way for somebody who knows JOREK well, but not your exact work. Not five pages, but briefly and precisely.
- **Reviewers should never have to reconstruct** what you are doing!

Various issues combined in one pull request

- **Reviewing becomes very hard**
- **Missing records about the changes**

- **Each issue needs to have its own jira issue and its own pull request.**
- This is a bit more work for you in the first place, but smaller pull requests get merged **quicker**, and usually end up **cleaner.**
- Separating out the issues has often helped to further **refine** the solutions, since there is **less confusion for everyone.**

Looks good, but does not work

- **When you raise a pull request, you should always test carefully what you have done. And not only for one special case.**
- In general it is your job to **convince the reviewers** of what you have done. This might require uploading a **derivation**, showing **simulation results**, etc.
- In addition, it can sometimes be a good idea if the **reviewers run special cases themselves** to actively test for possible problems, or if they **involve somebody else** who might have a critical application.

Assuming the other person is now in charge

- **Delays everything**

- **When you are not sure who is supposed to do the next step for a pull request, actively raise that question!**
- I try to always clarify the status of a pull request and **say explicitly who will do something next.** This is not meant as pressuring anyone or as telling someone what to do. It's only meant for **clarifying the status**. If you don't agree with what I say, **simply post a reply.**

Scattered discussions

- **Wastes time due to double-discussions**
- **Already accepted facts are forgotten again**

- **Jira and pull requests are the right place to raise points.**
- No-one will find discussions again if they take place partly in the pull request and partly via e-mail (and very often only between a subset of people).
- **Please avoid discussing development related things via e-mail** (only use them to point someone to a discussion in jira or a pull request)

Forgetting about already found issues in a pull request (or in general)

- **Wastes time**
- **Leads to merging code *we know* is not correct**

- When a further modification is agreed upon in a pull request, there needs to be a **“task”** created for it and closed when this is finished. Otherwise nobody can **keep the overview** of what is still open.
- And when something comes up in the pull request, that should be taken care of separately, a **jira issue should be created right away (direct link below the comment).**

Unnecessary changes across the whole code

- **Leads to avoidable conflicts with parallel developments**
- **Breaks “git blame” and makes reviewing a lot harder**

- **Pull requests need to have minimum changes necessary.**
- **Don’t change mod_parameters.f90 or the white space of existing code without a reason.**
- If you created a **“mess”** during your development, it might be best to **start again with a clean branch** and put only the necessary changes there. Don’t worry, has happened to me as well ;-)

Unreadable or unnecessarily cropped variable names

- **Makes the code hard to read and understand**

- **E.g.,**
  - **eta_spitzer is far more readable than eta_spi and can avoid future confusions**
  - **elm is not a good abbreviation for element**
  - **t can mean anything including time and temperature (remember Fortran is not case sensitive)**
  - **…**
- To find good names, put yourself into the position of **somebody else** stumbling across your code: **does she/he have a chance to grasp what the variable or subroutine is meant for from name/comments?**

Not the right level of commenting (none; incomprehensible; too detailed)

```
!> Extracts 1D boundary elements from 2D Bezier elements
!! see https://www.jorek.eu/wiki/...
subroutine xyz(a)

! --- routine parameters
Integer, intent(in) :: a     !< Number of …

! --- local parameters
real*8 :: r

! --- First, check which elements are…
do i = 1, … ! Loop over elements
…
end do ! i

end subroutine xyz
```

Unnecessary duplication of code

- **Leads to future inconsistencies and problems**

- **Generalizing a routine instead of duplicating is often very useful.**
- A great example is the **element_matrix and element_matrix_fft** routines, Stan recently combined into one for model303.
- In case of doubt, better discuss early. There are often elegant an less elegant solutions and maybe somebody you involve in the discussion has a good idea.

Inconsistent indentation, commenting styles, etc

- **Makes it hard to understand the code**

- **For *new* routines,** please follow the **coding guidelines** (in the wiki).
- **In *existing* routines,** stay **consistent with the "surrounding" code.**
- Only use blanks for line indentation, **never tabs**

No protection from breaking in the future

- **Works now, but *will* stop working at some point unless you take measures…**

- **When you implement a new feature, setting up a regression test should always be considered.**
- Otherwise, your feature can get **broken without noticing**. And experience shows that it will….
- Prepare a case with **minimal resolution** possible (just sufficient to run it, not physically resolve it). If it works, try again with lower resolution.
- **I can help you set up the regression test** based on the case you have prepared. That requires a simple description of how the case needs to be executed, some setup on the ITER Bamboo servers, and data to compare to (uploaded to jorek.eu).

Break the code when resolving conflicts

- **You might break your own code or someone else's**

- **While your pull request is open, develop might change since we merge other pull requests.**
- When merging these changes into your branch, **conflicts** can occur.
- **Be very careful resolving** those and ask for help if you need it. This step is a potential source for introducing problems. In case of doubt, **contact the author of the conflicting developments to resolve the problems together.**

Developments never merged

- **Your code is lost!!!**

- **You should always aim to get your developments into develop.**
- Please ask if you don’t know how. It’s **very sad if you invested a lot of time and your work dies silently** in a branch nobody knows about…
- **Only this way, we can all profit from the developments of the whole community.** Our code license in principle even obliges you to commit your changes to the git repository, by the way.

Unclear whether it is ready to merge

- **Can delays things, lead to merging to early, or cause unhappiness…**

Whether something is ready to merge depends on meeting four main criteria from my point of view:

1. **Correct and consistent?**
2. **Useful for at least some users?**
3. **Avoids affecting others unnecessarily?**
4. **Avoids affecting code structure and readability?**

The "meta-problem": Too long life-time of branches and pull requests

- **Caused by a combination of all the things mentioned before…**
- **Leads to conflicts with parallel developments, which need to be resolved. The longer the worse, usually.**
- **New features can't be used easily before they are merged.**

- **You are responsible for your pull request!**
  - **Discuss early via jira. Even before you start to implement.**
  - **Produce clean code. Only change what you need to.**
  - **Explain concisely what you are doing and convince the reviewers.**
  - **Keep the pull request up to date with changes in develop.**

All together…

… we have made **dramatic progress** with the development in my eyes.

… we can **further reduce problems and effort** for everyone if we follow the suggestions just described (or a refined version of them).

… we should never stop improving our workflows – if you have any **suggestions**, please bring them up!

If you have so far not touched jira, pull requests, etc, please start doing so. I am 100% sure you have found problems before and fixed them in your local version. Why not for everyone…?

What does this mean in practice

- **I might remind you of the “rules” in the pull requests**
- **I would regularly ask for the status of a pull request**
- **I would propose improvements**

- This is not meant as pressuring anyone, it’s an attempt to make everything work smoother and faster.
- You should obviously tell me in case you disagree with some of the things I’m raising in a pull request. We can then discuss and possibly ask for more opinions to find a common approach.

Your opinion on this?

Do you largely agree or do you rather disagree?

Do you have concrete proposals for improving?