Agile Project Initiation: Next stepping stones to transform us into an effective and productive agile team!

- Initial presentation of what agile development is
- what key points made out an agile team
  - knowledge sharing
  - different expertise joint to a powerful team
  - shared responsibilities
  - trustful and respectful team
  - get new team members up-to-speed by for example pair programming/reviewing etc.
- what are the benefits of an agile project planning
  - adaptive planning -> only specify things that are about to be implemented
  - keep stack holder in close loop about implemented features/changes to the project etc.
  - focus only on the important stuff
- Learned about several techniques to use to implement agile project planning/development
  - collecting user stories
  - working in small time units called sprints/iterations
  - Define stories that go into the release
  - Sprint planning meeting where the team discusses how to technically solve the issues, get feedback from stack holder what is required, what is the expected outcome -> define tasks for stories
  - Iterations planning meeting: Devs decide which stories/tasks (after they have been discussed in the sprint planning meeting) get into the next sprint/iteration
  - devs work on the defined tasks
  - making retrospectives to discuss the last sprint/iteration
  - make a demo for the stack holder and show the software

My experience and my current understanding:

- I am a library dev for 8 1/2 years now
- so far, I thought the library is the central key of this group -> better interfaces lead to bigger user base so I was very eager to get SeqAn3 into motion
- but: for bigger user base you need an outreach
- to convince people learning is worthwhile there need to be success stories
- ➔ **applications**
- in fact, the apps are the selling products, the business model
- the library is a mean to achieve this
- right now, we are targeting research areas, which are known to have no high-quality standards for the software they produce. So how do we change this? How can we convince them doing so is a great effort with multiplying effects?

I personal believe now:

- focus should be on the apps and their use cases
- from them derive the library design and implementation
- still design goal of the library: easy-to-use (stable) interfaces with stunning performance

Why do we need to transform to an agile team?

- goal for SeqAn3 was primarily to cover as much from SeqAn2 as possible
- several modules designed/implemented in parallel (1-2 persons)
- lead to ownership claim and skewed responsibility distribution
- instead of one ready module we have multiple open modules not fully ready yet
- designs and modules haven’t been validated through porting applications to SeqAn3
- it will lead to loss of design and put SeqAn3 into same situation as SeqAn2 before

Current team situation:

- 5 to 6 projects distributed on 8 devs and 2 SHKs as well as some external contributors
- 5 rookies
- 4 junior developers
- 1 senior developer

What happens if we work separately on the projects?

pro	contra
no need to justify	project takes long to get into an acceptable state (albeit we do not save time over all projects, since other projects will start later and finish later)
focus only on research	ownership claim; no shared responsibility
	more susceptible to design flaws
	different infrastructure for different projects
	scattered contributions to the library
	no benefit from different team roles
	missed opportunities for innovation

Why I think we can actually make this change

we already started to improve our working atmosphere -> and gained a lot of things from this:

- strategy meetings
- generated user stories for the different projects, regular project meetings
- weekly update meetings
- retrospectives
- topic specific iterations/sprints with the aim of collective code ownership

So, what do I want to change now?

From the experience we made and the things we observed in the last two years is that the team focus should be on one application at a time:

The first application we'll implement will achieve the following:

- design and setup of an application framework
- an infrastructure for spotting bugs, regressions between library versions and applications
  - a system for unit testing of applications
  - a system for micro benchmarking of applications
  - a system for macro benchmarking
- automatic packaging
- application site with corporate design
- designing library components based on the experience we made with the tool development
- possible pair programming with rooky/junior dev combination to get the team fully operational
- tutorials that target use cases and add therefor a great contribution to the community
- tackle company-wide performance regressions step-by-step
- start collective code-ownership
- designing and developing the library incrementally instead of bulky
- develop a clear definition of DONE for the team
- …

What are the perils:

- How much time are we investing in one project before we go to the next one?
- Does everyone will get his/hers share?
- What about pure research tasks and my PhD – do I have dedicated time to work on my personal research topic?
- Afraid of spending too much time in meetings discussing designs instead of actually implementing things and making progress
- Should every one become an expert for every feature/application/product?
- What are the risks of getting a feature not done in time and another project will fall behind its schedule with personal effects on single persons?
- Is the step we take too big at the moment and we should start more carefully especially given the risk that we don’t have a lot of experience with this project mode?

What project to start with:
 After evaluating the current project states as well as the planned teaching activities that lie ahead of us the best candidate would be the DREAM-Project

- already involves half the team
- has a lot of library functionality that needs to be added which is beneficial for most of the other projects
- was proposed to be taught on ISMB/ECCB etc.
- a lot of infrastructure will be created for all other projects to reuse
- we teach/implement the fundamental algorithmic approach currently developed inside of this group

One issue is the RNA-Aligner which is due at the end of this year.
We need to prepare the necessary steps for this as well.
before we can start we need to gather all deadlines and project obligations in order to plan this further.

What do I want to do with you in the next week then?

The inception deck:

- Why are we here? Identify the reasons for working on the project, and identify the one that is the most important.
- The elevator pitch. Articulate the business case for the project using a modified version of the user story format.
- Product box. Create a physical representation of a box for the product to help surface ideas.
- The NOT list. Identify what is in and out of scope.
- Your project community. Identify all of the stakeholders.
- Technical solution. Outline a technical solution or solutions, at a high level.
- What keeps us up at night. Identify the main issues/risks associated with the project.
- The A-Team. Identify the roles that are likely to be needed to complete the work.
- How big is this thing? Provide a very high-level estimate on how much work there is to do and how complex it is.
- Trade-off sliders. Indicate which areas are must-haves versus nice-to-haves, using a visual metaphor.
- The first release. Based on what is known, make some initial predictions about scope, delivery timeframe, and cost.

Decision:

The majority of the team wants to try this out and is in favour of changing the project planning. Especially the new team members see a great benefit in this. There are some perils that the team needs to be aware of and which need to be discussed thoroughly within the team.

It should be clear to everyone that the goal is not to change something just because of doing something different for a change, but to actually produce and maintain sustainable high-quality software without to many redesigns over the life-time of the future SeqAn, while at the same time every product/project is brought to a full success!
It is also about giving every team member the opportunity to become a full-fledged software engineer with a very broad skill set.