How can we as researchers at University learn from the workflow organization in tech-companies?

present: Jakub, Violeta, Moritz, Sven, Janina, Yousef, Maite, Johannes

last session's board: https://miro.com/app/board/o9J_lLAb_OY=/

Jakub and Maite summarized the following three articles for everyone:

Reinventing Research: Agile in the Academic Laboratory

https://www.agilealliance.org/resources/experience-reports/reinventing-research-agile-in-the-academic-laboratory

- problems in research lab: multiple projects in parallel, different people have different priorities w.r.t. the same project, skills are gained and leave the lab undocumented, knowledge silos means there is overlap in work/skills that people don't realise
- adjusting cadence of stand-ups depending on the project: once a day for quick small projects, weekly for long slow projects.
- visual workflow/whiteboard can be used if there are many projects that repeat a similar structure
- Difference with AGILE: "Working software over comprehensive documentation" does not work in the case of a laboratory, because comprehensive documentation is a key factor in tracking experiments. It's critical that an experiment can be replicated by other parties; a "working experiment" that cannot be reproduced is of no value.

How agile project management can work for your research

https://www.nature.com/articles/d41586-019-01184-9

- classic academic project follows the "waterfall" methodology
- an Agile PhD = break big experiment into a small one (2-12 week long);
- after it is done and analysed, add variables/conditions and merge with the old;
- add more data/variables/conditions, etc.
- include meetings with the stakeholders before and after each sprint.
- weekly supervision meetings focused on removing obstacles.

Scrum Of One: How to Bring Scrum into your One-Person Operation

https://www.raywenderlich.com/585-scrum-of-one-how-to-bring-scrum-into-your-one-person-operation

- Issue with being an independent developer: You're developer, SCRUM master and everything else all at once

- Getting organized to experience more productivity and satisfaction
- Key principles of Scrum to implement
    - ship and share, i.e. also show small improvements / results on a regular basis
    - priortizge productivity and quantify it - tracking and optimizing progress with Task Points
    - Self reflection and meaningful iteration - take a regular look at process and performance regularly
- How to Scrum for One
    - 1-4 weeks
        - Daily Scrum every morning 5 min
            - each day make a video that document yesterday's progress, today's plan and stuff that is blocking your productivity
            - Review yesterday's video, reflect on yesterday, plan for today and record
        - weekly Story Time 30-45 min
            - think big-picture
            - changing up location
            - review feedback from others, come up with concrete ideas for the Product Backlog, which you'll update at that time as well
        - Sprint Release
            - minor updates / results / …
            - not delaying feedback means also prioritizing most important features
        - Last Day of Sprint
            - retrospective, clean up, plan next sprint, do sth. fun
        - Retrospective 2h
            - thoughts about past two week's productivity
            - what accomplished? sprint goal met? what could have been better? Review Daily Scrum videos, what was stressful and nice? what to improve next time - what could make you more productive, what happier?
        - Sprint Plan at the end 2h
            - use output from Story Times, Backlog and Retrospective for input about next sprint, set sprint goals, tasks, and task points
            - Task Points:
                - One-Point Task half as two-point task, perhaps only use 1,2,3,5,8 to not underestimate hard tasks
                - use for reasonable predictions about how long YOU will take doing that
                - add up points, compare to what was achieved in last sprint / day
            - Task Board (organize end of the day for the next day)
                - day to day task organization (Kanban: To do, doing, done)
                - start the day by moving one task to doing and work only on that, when done enjoy moving to done ;-)

Group discussion:

- the main problem we have with using AGILE is that there is very different topics on our plate each week, there's no way focusing all attention on one sprint only (i.e. teaching, admin stuff, research)
    - could be helped by combining with time blocking system
- Is there any software that could be used well for organizing your workflow?
    - github issues as documentation of tasks / problems
    - the analog version - paper, markers and sticky notes
    - miro board
    - Trello
    - checklists for almost everything and see the progress bar
    - keep track of what you did
    - pomodoro 25 min chunk
    - perhaps it's not necessary to have a specific technology for specific methods but rather the concept is important, the implementation can vary a lot
    - depends on your mood as well what you want to use at that time
    - Attention - you can easily spend more time organizing your workday than actually working! Keep it minimal
    - cross off what you already did - it's a good thing to see how your hours and tasks pile up during the day
    - physical and digital system setup at the same time - conclusions of the week, tasks per day, if it doesn't get done, explain why
        - google calendar color coding for different sort of tasks, i.e. meeting / data analysis / studying / something important
        - 2h tasks
        - in the end of the week review whether the main emphasis was actually on the most important category
    - humans tend to select smaller tasks, stuff that has deadlines
        - take a small task from the big task, instant reward, little quick rewards are very important
        - we tend to forget what we did - documentation important for keeping up good work
    - how to make sure that other people can pick up your work?
            - using github
            - make a readme
            - good documentation / structure of your code
        - for explaining why you chose certain requirements / assumptions in your workflow, good code may not be enough, but documentation needs to be updated
        - block course on reproducibility AGILE reproducibility coming up (ask Jakub for more info or what you're interested in specifically)
        - O2R team (Daniel Nüst) can give you feedback on the reproducibility of your code
        - tests that are executable in an automatic way
            - Does it make sense only for software programming or analysis as well?

Resources shared via the Chat:

https://twitter.com/AcademicChatter

QUIZ: Which productivity method is right for you? https://todoist.com/productivity-methods

how to organize your home office day and stay focussed:
https://www.youtube.com/watch?v=AgIggrkBFcU

https://www.youtube.com/watch?v=zwRdO9_GGhY
(Science as Amateur Software Development) - on version control, testing, etc.
in the research context

The A-Z of the PhD Trajectory: A Practical Guide for a Successful Journey Lantsoght, Eva O.L.
https://www.springer.com/gp/book/9783319774244

Stuff we had foreseen to discuss but ended up using only partially - e.g. possibly useful for upcoming discussions

Which problems do we potentially face that could potentially be solved by using AGILE?

- being overwhelmed by vastness of a problem (I'm thinking use cases and sprints)
- juggling multiple projects / teaching etc.
- keeping yourself on track / procrastination / accountability / staying motivated
- have your work organized in a way that other people or yourself can use it quickly (again)
- how to fail fast & move on

Implementation

- How would it look in practice, is it viable for each of us?
- How much working hours would it cost us to implement it?