Focus group 6 - Software workshop 16 Jan 2017

Problem(s) that the group wants to solve:

How to effectively work with technical and non-technical people when developing software?

Facilitator: Kirstie Whitaker

Rapporteur:

Kirsten Lamb

Participants:

- Kirstie Whitaker
- Anne
- Justin
- Matt
- Jason
- Michael
- Martin
- Marta
- Sarah
- Laurie
- Kirsten
- Adam

Notes:

- Challenges of working with a mixture
    - Technical people don't understand biology
    - Non-technical people don't understand technical things, mostly
        - Language is an issue/jargon
        - Even amongst the technical team people need to agree meanings before starting a project.
            - But is this a solution, to sit down at the beginning of a project and agree how terms work?
                - Support for this idea.
                - But can people write and understand a specification?
                - Idea that this is an exercise that is repeated - people learning these terms and to understand each other over time.
                    - Iterative development, start small, give prototype, not building walls, tearing them down.

        - But lab with computational and wet lab people all in one room doesn't work well, with computational people need to concentrate for longer periods of time compared to wet lab. If you hold meetings all together then you get 3 people at the front of the room talking all the time whilst everybody else is silent.
        - Not trained in it. Support for this idea of miscommunication between 2 sides. Both sides assume that the other knows more than they do (assumptions) without making these explicit. One of the most dangerous moments is when there isn't quite enough knowledge. Maybe need more training on people from different fields and cultures communicating with each other.
        - One thing that reveals this is to ask what people thought a talk was about after they heard it. Then 2nd iteration of fixing these issues
        - Can't assume that a collaborative project goes on like that all the time. Need to have consencus at key points in the project. But, still people need to talk to each other.
    - Different domains of knowledge in fast moving fields.
        - Have to keep repeating scenarios to get concensus at what a technical team can actually deliver for non-technical people.
    - Funding limits
        - How do you square the resource needs with the planning.
    - Technical people will contradict/not understand the non-technical parts of the field (e.g. analyzing a dataset in flow cytometry).
    - When technical and non-technical work together, is one serving the other? Is one reach out to the other to achieve a goal, or is there a shared goal? Maybe they both think they are acting for their own goal (technical ppl: produce tool, biologist: do science on a particular dataset).
        - Why are they working together? To produce something bigger than the sum of the parts.
        - 5 years ago you could have said tech was supporting science, but now these things are becoming circular, technical and non-technical rely on each other.
            - Data science coming in as a 3rd cog. They deal with complex modelling and analysis, scientists the ideas, engineers the software.
            - Another example: core facility deal with the public and collect samples, the grunt work. But they do not understand what is required for analysis in flow cyclometry. With increasing computational work, the line between data collection and lab

work and computation is becoming blurred. E.g. opening a data file, information may not be labelled well - the person doing the data collection does not realize the importance. Would be really helpful to have someone in the middle who understands what both sides require. SOPs are not working.
        - Computational people may consider data collection dogsbody work, this can be extremely frustrating (e.g. get a child to sit in an MRI scanner for an hour).
    - Need to be able to have people come into the field who can stick to their disciplines without bridging everything.

Quick wins:

- Sample problem - writing a paper collaboratively, use online tools instead. But there is inertia, so many tools, comfort in the status quo. Overleaf, google docs.
- Iterative development. More awareness of stages in the life cycle. Possibly less emphasis on talking continuously (classic agile in software) but have checkpoints where people come together and understand things.
- Doc, doc, doc. Check, check, check. Test, test, test.
- Technical and non-technical people need to learn skills both ways. Both need to have knowledge of the other side. They need to be able to communicate (tell people what they are doing and how they work). This needs time and money, support from PIs, management. Having courses available helps (Cambridge is good for this).

Long-term solution(s):

- Collaborations to have a set number of experiences that every new member of the team would go through. So that everybody has some basic exposure.

Summary: