So, You Want to be Agile?
(the slides are available under “Presentation Materials” in the above URL)
Date: May 08, 2019
Presented by: Michael Heroux (Sandia National Laboratories)

Q. Is there one tool for agile? Or I need more than one?

A. Most people would want to use more than 1 tool. Common toolset is Atlassian tools: you get Confluence (online collaborative wiki for lots of different types of content). Sandia uses Confluence with positive impact. Another Atlassian tool is Jira for issue management. Also very nice tool. Jira more complicated than other issue tracking tools - might not be a good choice of first tool. Atlassian recently purchased Trello. Most teams use Confluence and Jira - good infra for Agile practices of scrum and kanban. Another toolset is GitHub and GitLab. Both offer similar wiki documentation, issue tracking, lighter weight project boards for scrum or kanban.

Q. Would you consider “kanban” a subset of Agile? Or simply a stepping stone?

A. Because we have so many other demands on our time and our work is interrupt driven, it’s very difficult to do a formal sprint. Smaller teams; not clear who the product owner or scrum master is. Having a coach for the methodologies helps. Don’t know if kanban is a step towards scrum; it could be? Scrum has many positives but has a lot of responsibilities.

Q. Can you briefly explain the new Software Scientist role again?

A. We want to understand how to make better scientific software. How do you make developers more productive? How do you make scientists more productive? Turn it into a science problem. We know how to study human systems. We can see correlations between connectedness and happiness - that was an outcome of the paper I arxiv showed you. That’s just correlations - turn it into a science question: does connectedness lead to happiness? Take social science tools and apply them to scientific software. (Thanks for explaining! <3)

Q. How big a team needs to be before all these extra efforts become worthwhile?

A. Kanban is useful for a team of 1! I use it for managing my personal tasks. Grows in value as you have more team members. Also grows in values with distributed tools and distributed teams. I work a lot with students. The literature I've read says scrum requires at least 5 people to see the value from it and less than 9. For more than 9 you need "scrum of scrums" hierarchical structure.

Q. With the advent of using your concept of a computer scientist role in project would the CS portion be more Scrum based and the science end be Kanban based?

A. I haven't thought about that! Attractive idea worth thinking about.

Q. Could you send out the arxiv citation again?

A. https://arxiv.org/pdf/1809.06317.pdf

Notes:
Slide 4 - Intro:

- Will recap kanban.
- Better planning.
- Team techniques and introducing new practices and tools.
- New notion of "Software Science"

Slide 6

- http://agilemanifesto.org/ - things on the left in larger font are more valuable.
- Slide 7: we're too focused on things on the left, and not enough on things on the right, especially "customer collaboration".

Slide 8

- Fred Brooks of the "Mythical man month". Science does barely sufficient building, because goal is insight / learning. But for an engineer the goal is building.

Slide 10

- Scrum (most popular in industry, ~80% of team), kanban, extreme programming
- Scientific software developers spend a lot of time in the community. We keep our PhD degree fresh by keeping up with the field; little time for cultivating advanced software practices.
- Scientific devs tends to have specific expertise; often only one person can complete a task. Adopting agile approaches should account for this.

Slide 14

- Checklists and policies
- Checklists - new team member checklist and departing fellows checkist. Master checklist to ramp up new students and measure progress.
- In middle of checklists are policies to build team understanding and expectations for behavior. What best practices are expected? [Audio has cut out :(...]

Slide 15

- Industry partners of Exascale project say scrum is very effective. Works in methodical and steady life. Most teams practice “scrum, but…” where they eliminate some things from scrum.

Slide 16

- Kanban easier to take on than scrum. Scrum has a harder time promoting discovery work. Kanban generally easier.
- Limits tasks in-progress.

Slide 17

- Backlog -> Ready -> In Progress -> ***Blocked*** -> ***In Review*** -> Done
- “Blocked and “In Review” are Michael’s additions ;)

Slide 18 - Planning

- Eisenhower: Plannning process is most important; try in your head approaches to solve the problem. Then find what’s relevant from exploring other options and adapting and benefiting from planning activities.

Slide 21

- Code-and-fix approach
- Didn’t think about the design of the software; just a notion of what it should do.
- Time gets sunk because of not thinking of the requirements, architecture, platforms, etc.
- Can spend so much time in re-coding and porting that you can’t use the code anymore!

Slide 22

- Upfront time investment in design and planning. Don’t get early visible progress in code, but helps in the long run.

Slide 24

- How does one collect requirements? What practices do we use? One is using stories.
- Stories follow a pattern.
- 2 types of stories
    - User stories: “As a ____ I want ____ so that ____”
    - Job stories (more natural to our Sandia teams): Activity rather than role based. Fewer users and active users, but more sophisticated activities.
- Stories help you prioritize tasks.

- Stories work for some situations. Need skill and adaptation. Not a prescriptive technique! But can be valuable to teams.

Slide 25

- Phase 1 is generating stories.
- Collect table of stories in a shared document.

Slide 26

- Phase 2: out of scope? Everyone should understand what each phrase means. Size / scope them to be accomplished in a reasonable amount of time.
- Prioritize the top set of stories, not necessarily all of them.

Slide 27

- Stories useful in tool selection, not just software development.
- E.g. picking IDE for students. Coming from Java to C++. One team stumbled on AWS Cloud 9 - the Google Docs of IDEs. Came out of exploration phase of using stories.

Slide 29 - Documenting design

- Making design mistake in document is easier to spot and change. Finding flaws in software is much harder and longer to fix.
- Lots of tools available. Don't need to learn UML, class diagrams and interaction diagrams. Instead simply write up what you will produce as a piece of software.
- E.g. Kokkos Kernels.
    - Took 6 weeks to write LaTeX design document,
    - Passed the document around to experts in architecture and design development, users.
    - They made very significant changes!
    - Found moving from CPUs to GPUs the design was not optimal.
    - Programming models and architecture expertise allowed them to get expertise they would not have had without documentation work.

Slide 31 - Strategies for introducing change

- We don't send our devs out for months-long training.
- We need to incrementally improve (e.g. PSIP-Tools).

Slide 33* - example unit test card e.g. for MPICH team

Slide 34

- MPICH lightweight onboarding card

Slide 37 - Appealing to each person on the team for ownership and self-improvement

- Often GitHub stats heatmap of commits is the wrong metric.
- Undisciplined code commits of undocumented, poorly archtected code base adds more technical debt.
- Instead recognize people who write up requirements, analysis and design.
- Good issue filing. Good responses to users.

- Better metrics and more holistic than counting commits to repo.

Slide 38 - Productivity++ card

- Tracable = linked to an origin that work is important
- In Progress = Planned activity in kanban? Or (bad) from phone call or personal prod?
- Sustainable = proud of the work and should life long
- Improved = I improved myself to do this work.

Slide 40

- Science as opposed to the engineering of software, called SEAR at Sandia.
- Trying to develop and model in an organization setting.
    - Deployment = IT type person
    - Research Software and Engineering = Dev
      Research Phase = new workflow!

Slide 41

- People work across these areas.