Week 7 Workshop

Sprint review

Objectives

In this week's workshop, you will:

- conduct a sprint review to demonstrate your product to your client (i.e. demonstrator)
- conduct a backlog refinement session
- plan or conduct a retrospective

Task 1. Sprint review (for project marks!)

This week, you need to demonstrate your product to your facilitator, who will mark you on your progress. Other parts of your sprint review can be done without your facilitator's presence, but they must be present for the demo. Your facilitator will organize a time for you to conduct this demo. They will also need to review the products of all other teams in your class, so time might be limited.

Be ready to explain which user stories your team has gotten Done, and to demonstrate that they work. Have your Product Owner pay close attention to any feedback that your team receives – if your team has misunderstood the client's needs or priorities, you might have a lot of work to do in backlog refinement.

Task 2. Backlog refinement

Following the description in the Week 6 compulsory readings, conduct a backlog refinement session to get your backlog ready for your next sprint. Add any new stories you've discovered, and make sure that there are a few stories that are ready to be selected into the next sprint backlog. You may begin your next sprint planning session as soon as your sprint review and retrospective are finished, and you can combine it with backlog refinement if you have the time.

If you're worried that your marker might get confused by partially-implemented Sprint 2 stories in your codebase, we suggest you create an *annotated tag*. You can do this through the gitlab UI[1]. Go to the page for your repository, click Tags in the left sidebar menu, click the green New Tag button, then give your tag a name and add a message. Put a note in your `readme.md` file to tell your marker which tag they should mark.

Task 3. Retrospective

You should use the remainder of your class time to either conduct or plan a retrospective meeting. This is especially important if the product review didn't go well – if your velocity is unacceptably low, your team needs to consider how you might change your practices to improve it in future iterations.

Have somebody take notes on your retro and place them alongside the other documents in your git repository. These notes don't have to be written up formally, but should make it clear what changes your team has decided to make. A good retro can at least partially compensate for a poor sprint review outcome if the problem was caused by poor practices rather than by technical inadequacies.

[1] You can probably do it via your preferred git GUI as well – go read the docs. To create a tag using the command line version of git, type `git -a annotation -m message` – for example, `git -a Sprint1 -m "End of sprint 1"`